Rechnergestütztes Baugruppenprüfgerät zum universellen Test verschiedener Baugruppen von Röntgendiagnostikgeräten

Wolfgang Schraml

WS 1989/90



FACHHOCHSCHULE DER DBP DIEBURG

DIPLOMARBEIT

von

Wolfgang Schraml

REFERENT: Dr. Metzendorf
DIEBURG
WS 89/90

THEMA

der

DIPLOMARBEIT

Rechnergestütztes Baugruppenprüfgerät zum universellen Test verschiedener Baugruppen von Röntgendiagnostikgeräten

Folgenden Personen und Firmen möchte ich im Zusammenhang mit der Bearbeitung meiner Diplomarbeit danken:

- Hr. Diplom-Ingenieur (FH) Gollwitzer für die beispielhafte Betreuung meiner Arbeit bei der Siemens AG

- Hr. Dr. Metzendorf, der mir als Ansprechpartner und Betreuer an der FH Dieburg jederzeit mit Rat und Tat zur Verfügung stand

- der Firma Siemens AG, Bereich Medizinische Technik, Werk Kemnath, die mir freundlicherweise ihre Geräte und Einrichtungen zur Verfügung stellte

1. Einleitung

Die Abteilung Qualitätssicherung der Siemens AG, Unternehmensbereich Medizinische Technik, in Kemnath steht vor der Aufgabe, eine Vielzahl verschiedener elektrischer Baugruppen von Röntgendiagnostikgeräten auf ihre Funktionsfähigkeit hin überprüfen zu müssen. In früherer Zeit wurde dieses Problem dadurch gelöst, daß für jede zu prüfende Baugruppe ein eigenes Prüfgerät entwickelt und gebaut wurde. Da die Vielfalt der Baugruppen immer größer wurde, stellte man Überlegungen an, ein Prüfgerät zu entwickeln, welches universell einsetzbar sein sollte. Das Ergebnis dieser Überlegungen war der sogenannte WÖTRA A, ein universelles Baugruppenprüfgerät. Gesteuert wurde es durch einen Commodore 8032 Rechner, auf welchem auch die verschiedenen Prüfprogramme in Commodore Basic verfaßt wurden. Im Laufe der Zeit traten jedoch einige schwerwiegende Mängel, wie z.B. hohe Empfindlichkeit gegenüber Störungen von Außen und umständliche Programmierung aufgrund des doch sehr primitiven Basic-Interpreters des Commodore-Rechners, auf. Aus diesem Grunde wurde das Konzept des WÖTRA B, eines auf PC-Basis arbeitenden, störsicheren Prüfgeräts, entwickelt.
Es wurde noch eine weitere Neuerung mit in das Konzept aufgenommen: während früher für jede Baugruppe ein extra Prüfprogramm erstellt werden mußte, in welchem die Ansteuerung des Geräts und die Auswertung der Messungen durchgeführt wurden, ist es in der jetzigen Ausführung so, daß mit Hilfe eines Prüfprogrammgenerators für jede Baugruppe eine Prüfdatei erstellt wird, in der die einzelnen Prüfschritte abgespeichert werden. Diese Prüfdatei wird dann von einem Prüfprogramm ausgewertet, daß dann in jedem Falle gleich bleibt. Dieses Prüfprogramm stellt also eine Art Gerüst dar, welches erst unter Zuhilfenahme der entsprechenden Prüfdatei zum speziellen Prüfablaufprogramm für die zu testende Baugruppe wird.

Das gesamte Projekt WÖTRA B wurde auf insgesamt drei Diplomarbeiten aufgeteilt. Der erste Teil, die Entwicklung der Hardware, d.h. der Ansteuerungs- und Dekodierlogik wurde von Stefan Seitz, Fachhochschule Regensburg, bearbeitet. Der zweite Teil, die Entwicklung des Prüfprogrammgenerators, war Aufgabe von Markus Kölbl, Fachhochschule Nürnberg.

Meine Aufgabe innerhalb des Gesamtkonzepts war es, den mechanischen und elektrischen Aufbau des WÖTRA zu überwachen sowie die Inbetriebnahme durchzuführen. Dies schloß auch die Endmontage der verschiedenen Leiterplatten, Stromversorgungen und Kabelbäume im Gehäuse des Prüfgeräts mit ein. Ferner war ich zuständig für die Entwicklung der übergeordneten Benutzeroberfläche in Window-Technik, der verschiedenen Testprogramme, des eigentlichen Prüfablaufprogramms sowie des Ansteuerprogramms für das IEC-Bus gesteuerte Meßgerät. Während der Überwachung des Aufbaus entwarf ich auch noch die verschiedenen Umschaltplatinen des WÖTRA B.

Ferner wurden noch Versuche zur galvanisch getrennten Überwachung der verschiedenen Prüf- und Versorgungsspannungen des WÖTRA B durch den Steuerrechner durchgeführt.

2. Beschreibung der Hardware

2.1. Allgemeines

Damit das Prüfgerät möglichst universell einsetzbar ist, stellt es dem Benutzer 400 Anschlußpunkte, im folgenden auch als Knoten oder Koppelpunkte bezeichnet, zur Verfügung. Jeder dieser Koppelpunkte ist entweder als Ein- oder Ausgang programmierbar. Eingang bedeutet in diesem Zusammenhang, daß an den ensprechenden Koppelpunkt ein IEC-Bus gesteuertes Meßgerät oder ein A/D-Wandler-Eingang des Steuerrechners angelegt werden kann. Der A/D-Eingang dient dazu, um eine schnelle Durchgangsprüfung realisieren zu können, das Meßgerät ist für die Messung von Gleich- und Wechselspannung, Gleich- und Wechselstrom sowie für Widerstandsmessungen zuständig. Wird ein Koppelpunkt als Ausgang programmiert, so kann an ihn eine der folgenden Prüfspannungen angelegt werden: 220V AC, 110V AC, 24V AC, 24V DC oder eine variable Gleichspannung im Bereich von 0-100V. Kernpunkt all dieser Verschaltungsmöglichkeiten ist ein Koppelfeld von 2000 Relais, welche sich auf insgesamt 100 Relaiskarten im Europa-Format befinden. Die 20 Relais je Relaiskarte sind jeweils vier Knoten zugeordnet, d.h. fünf Relais pro Knoten oder Ausgang. Die Schaltkontakte der fünf Relais sind auf der einen Seite zusammengefaßt und an den jeweiligen Ausgang geführt, auf der anderen Seite führen sie zu je Einer von fünf sogenannten Relais-Bus-Leitungen. Dieser Relais-Bus ist über alle 100 Relaiskarten durchverbunden, so daß also jeder Ausgang des WÖTRA wahlfrei auf eine dieser Busleitungen geschaltet werden kann. Durch eine weitere Umschaltung ist es möglich, auf diesen Relais-Bus entweder das Meßgerät bzw. den A/D-Wandler Eingang oder die verschiedenen Prüfspannungen zu schalten. Dadurch ist dann gewährleistet, jeden Ausgang des Prüfgeräts wahlfrei beschalten zu können.

Die Ansteuerung der einzelnen Relais geschieht folgendermaßen: Ein Anschluß aller 2000 Relais-Erregerspulen ist fest auf +24V Gleichspannung gelegt. Die jeweils anderen Anschlußpunkte sind auf Steckerleisten geführt, welche durch einen Kabelbaum mit den Dekodierplatinen verbunden sind. Soll nun ein Relais eingeschaltet werden, so wird von der Dekodierplatine einfach die Masse der +24V Versorgung durchgeschleift, so daß das Relais anziehen kann.

Die Beschreibung der Ansteuerung und Dekodierung soll hier nur einen groben Überblick über die Funktionsweise verschaffen. Eine detailierte Darstellung der Zusammenhänge mit den ausführlichen Schaltplänen befindet sich (2). Lediglich der Schaltplan der Relaisplatinen wurde im Anhang mit aufgenommen, da er für das Verständnis des später beschriebenen Selbsttests sehr wichtig ist.

2.2. Die Ansteuerung

Die Ansteuerung des Prüfgeräts WÖTRA B erfolgt durch einen AT-kompatiblen Rechner Siemens PCD-2, der folgende Daten und Erweiterungen aufweist:

- CPU 80286
- 1 Mbyte RAM Haupspeicher
- 20 Mbyte Festplatte
- Betriebssystem MS-DOS 3.2
- VGA kompatible Grafikkarte
- VGA Farbmonitor
- universelle Portkarte mit 24 digitalen Ausgängen und 16 analogen Eingängen der Firma Impec
- IEC-Bus Schnittstellenkarte der Firma Philips

Die 24 digitalen Ausgänge der universellen Portkarte dienen dazu, den WÖTRA-Bus, d.h. den internen Bus des Prüfgeräts,

(2) siehe Diplomarbeit Stefan Seitz

zu generieren. Dieser Bus besteht aus 15 Adreßleitungen, 8 Datenleitungen und einer Write-Leitung. Die 15 Adreßleitungen werden benötigt, um jeden der 400 Ausgänge ansprechen zu können, d.h jedem Ausgang ist eine feste Adresse zugeordnet. Von den 8 Datenleitungen werden fünf dazu benutzt, jeweils das gewünschte Relais des ausgewählten Ausgangs anzusprechen, die drei Übrigen sind für eine spätere Erweiterung des WÖTRA auf acht Relais-Bus-Leitungen vorgesehen und im jetzigen Zustand nicht verdrahtet. Die Write-Leitung legt fest, zu welchem Zeitpunkt die am WÖTRA anliegenden Adressen und Daten gültig sind.

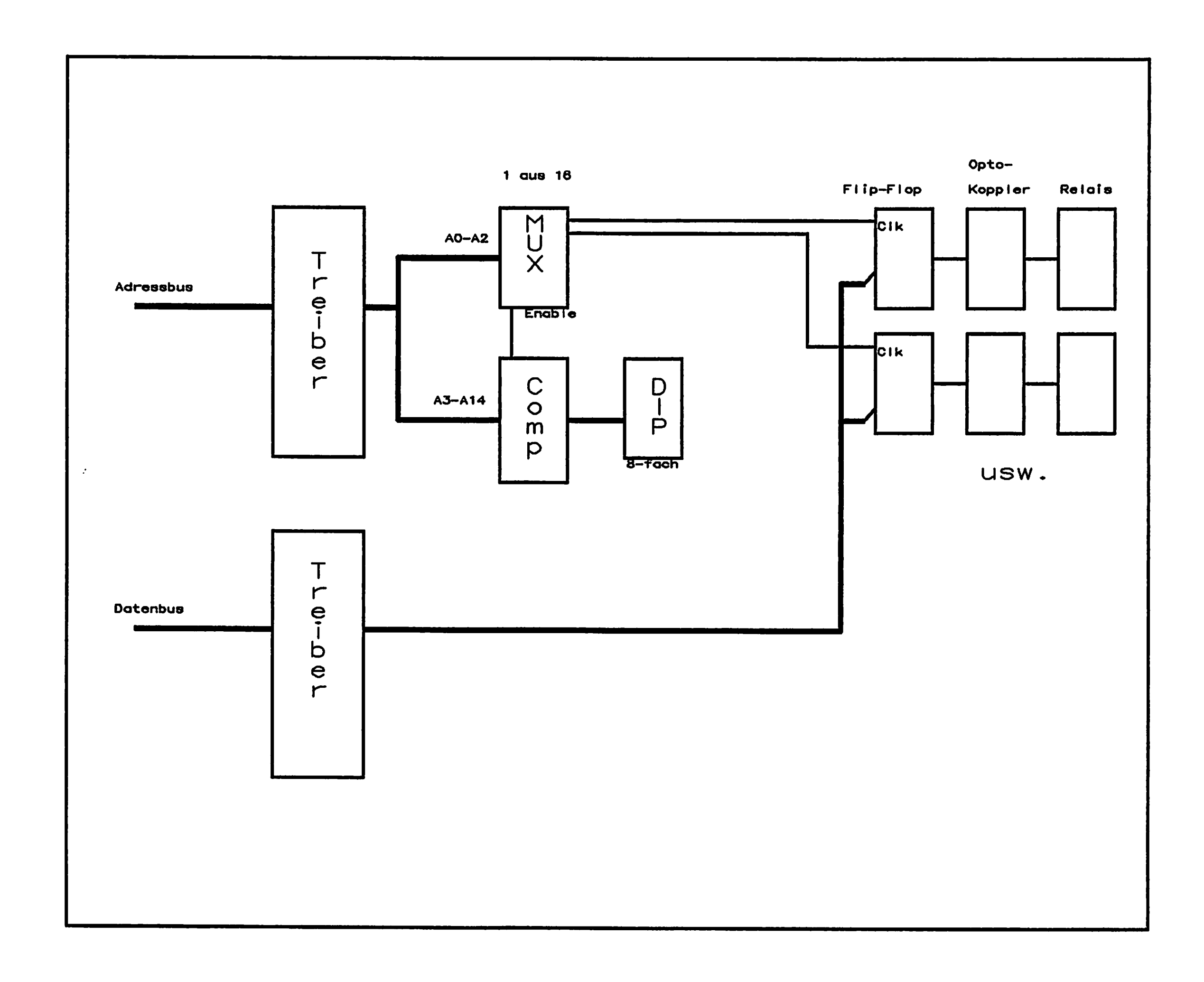
Die Ausgänge der universellen Portkarte werden zunächst auf eine Treiberplatine geführt, welche drei ICs vom Typ 74 LS 240 beinhaltet. Diese Treiber sind nötig, um die nachfolgenden Optokoppler auf der Isolationsplatine ansteuern zu können. Um ein Höchstmaß an Störsicherheit zu erhalten, wurde die Treiberplatine in eine HF-dichte 19-Zoll-Einschubkassette der Firma Schroff eingebaut. Dies mag auf den ersten Blick überzogen erscheinen, hat aber aufgrund der Entwicklungsgeschichte des WÖTRA-Konzepts seine Berechtigung. Bei der früheren Auslegung des WÖTRA auf Basis des Commodore-Rechners wurden keine besonderen Maßnahmen zur Erlangung von Störsicherheit durchgeführt. Dies rächte sich aber bald durch die Tatsache, daß sehr oft beim Schalten einer Relais-Baugruppe Störspitzen in den Rechner gelangten und diesen zum Totalabsturz brachten. Solchen Vorkommnissen sollten beim jetzigen Gerät von Vornherein keine Chance gegeben werden.

Alle Ein- und Ausgänge der Treiberplatine sind zusätzlich noch mittels Durchführungs-Enstörelementen durch das Gehäuse der HF-Kassette geführt.

Vom Ausgang der Treiberplatine führt dann ein Kabel zur sogenannten Isolationsplatine DW2, die dazu dient, einen galvanisch getrennten Übergang von der 5V-Logik des Rechners zur 15V-CMOS-Logik der Dekodierplatinen zu schaffen. Die 15V-Logik wurde verwendet, um einen möglichst hohen Störspannungsabstand zu erhalten. Dieser beträgt bei

der 15V-CMOS-Logik ca. 6,75V, bei den 5V Standard TTL-Schaltungen, welche im alten WÖTRA eingesetzt wurden, dagegen nur ca. 1V. Diese höhere Störempfindlichkeit äußerte sich dann dadurch, daß oft ohne ersichtlichen Grund ganze Reihen von Relais anzogen und Kurzschlüsse auf den Relais-Bus-Leitungen verursachten. Eine erhebliche Störung des Prüfablaufs war die Folge.
Der Übergang zwischen den beiden Logikpegeln ist auch deshalb durch Optokoppler verwirklicht worden, um eine galvanischen Trennung zwischen Rechner und WÖTRA zu erhalten. Dies trägt wiederum zu einer Erhöhung der Störsicherheit bei. Die Ausgänge der Isolationsplatine führen auf zwei Busplatinen mit 96 pol. VG-Steckverbindern (Siemens SMP-Busplatinen), auf welchen auch sämtliche Dekodierplatinen gesteckt sind.

2.3. Die Dekodierung



Die Adreßdekodierung des WÖTRA B ist auf insgesamt 25 Platinen des Typs DW3 (1) im Doppel-Europaformat untergebracht. Jede dieser Platinen ist in der Lage, vier Relaiskarten und damit 16 Ausgänge anzusteuern. Alle Dekodierplatinen erhalten an ihrem Eingang die 15 Adreß- und 5 Datenleitungen sowie eine Write-Leitung. Alle ankommenden Signale werden zuerst einmal durch drei Treiber vom Typ 40244 verstärkt. Die Adreßleitungen A0 bis A3 führen dann auf einen 1 aus 16 Dekoder vom Typ 4514, welcher dann je nach anliegender Adresse einen der 16 Ausgänge freigibt. Die höherwertigen Adreßleitungen werden zusammen mit einem 8-fach DIP-Schalter zwei 4-Bit Komparatoren (4585) zugeführt. Diese Verschaltung dient dazu, die Basisadresse jeder Dekoderplatine mit Hilfe der DIP-Schalter festlegen zu können. Der Vorteil dieser Art der Adreßfestlegung besteht darin, daß alle Dekoderplatinen vom Aufbau her gleich sind.
Die Ausgänge des 1 aus 16 Dekoders sind anschließend mit den Clock-Eingängen der ICs 40374 verbunden. Diese Bausteine enthalten jeweils acht D-Flip-Flops mit nachgeschaltetem Treiber. Die Dateneingänge der Flip-Flops sind mit den Datenleitungen des WÖTRA-Bus verbunden, die Ausgänge führen auf die Eingänge von nachgeschalteten Optokopplern, welche die Aufgabe haben, den Übergang von der 15V-Logik zur 24V Versorgungsspannung der Relais zu schaffen. Außerdem sorgen sie durch ihre galvanische Trennung von Ein- und Ausgang nochmals zu einer Erhöhung der Störsicherheit.
Wird nun eine Adresse ausgegeben, so wird durch den 1 aus 16 Dekoder eines der Flip-Flops freigegeben und die anliegenden Daten zum Eingang des folgenden Optokopplers durchgeschaltet. Dieser steuert ebenfalls durch, worauf das nachfolgende Relais schaltet kann.

(1) siehe Diplomarbeit Stefan Seitz

2.4. Die Umschaltung

2.4.1. Allgemeines

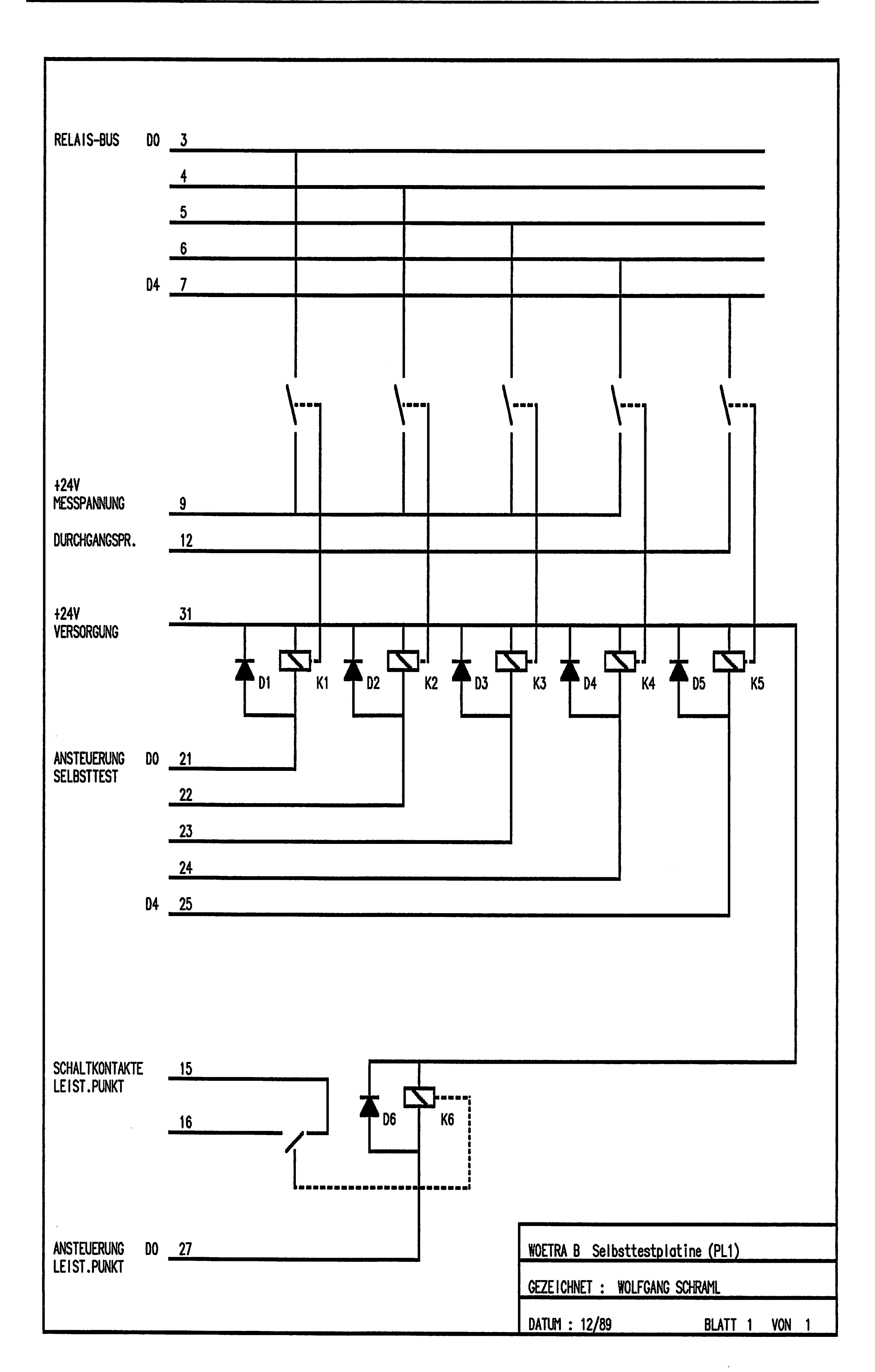
Bei der jetzigen WÖTRA-Version sind lediglich fünf Relais-Bus-Leitungen vorgesehen, um kompatibel zu den bisherigen Geräten zu bleiben. Da jedoch fünf verschiedene Prüfspannungen, ein Meßgerät und ein A/D-Wandler-Eingang zur Verfügung stehen sollen, ist es notwendig, je nach durchzuführendem Prüfschritt die verschiedenen Spannungen und das Meßgerät auf die Relais-Bus-Leitungen zu schalten. Dies wird durch fünf Umschaltplatinen bewerkstelligt, welche nachfolgend im Einzelnen genauer beschrieben werden sollen.

2.4.2. Selbsttestplatine

Um die Funktionsweise der Selbsttestplatine zu verstehen, ist es zunächst einmal erforderlich, etwas über die Durchführung des Selbsttests zu sagen. Er wurde ins WÖTRA-Konzept mit aufgenommen, um die Relais des Koppelfeldes dahingehend überprüfen zu können, ob sie eventuell beim Schließen keinen Kontakt geben oder beim Öffnen kleben.
Um dies möglichst einfach und schnell zu ermitteln, wird folgendes Prinzip verwendet: die 400 Ausgänge des Prüfgeräts werden paarweise kurzgeschlossen, d.h. Ausgang 1 und 2, Ausgang 3 und 4, usw. bis Ausgang 399 und 400. Hardwaremäßig wird dies durch Kurzschlußstecker an den Ausgangsbuchsen des WÖTRA realisiert. Anschließend wird auf die oberen vier Relais-Bus-Leitungen (siehe Schaltplan der Relaiskarten im Anhang) die +24V Prüfspannung gelegt, auf die Untere (die sogenannte Meßleitung) der A/D-Wandler-Eingang zur Durchgangsprüfung. Wenn nun das Relais des ersten Ausgangs, daß auf der einen Seite mit der Meßleitung verbunden ist, eingeschaltet wird, liegen automatisch alle Relais der beiden kuzgeschlossenen Ausgänge mit einer Seite auf der Meßleitung. Werden nun die Relais des zweiten

Ausgangs der Reihe nach durchgeschaltet, so muß am A/D-Wandler-Eingang jedes Mal die 24V Prüfspannung anliegen, da ja die oberen vier Relais-Bus-Leitungen mit dieser Prüfspannung beschaltet sind. Ist dies nicht der Fall, so schließt entweder das entsprechende Relais nicht richtig, oder es liegt ein Fehler in der Verdrahtung der Ausgangsbuchsen vor. Analog dazu muß beim Ausschalten der Relais die Prüfspannung wieder verschwinden. Tritt dies nicht ein, so "klebt" das entsprechende Relais, d.h. es hat immer Kontakt, auch im erregungslosen Zustand. Sind nun alle Relais des zweiten Ausgangs auf diese Weise geprüft, so wird nun genau umgekehrt verfahren, d.h. beim zweiten der beiden kurzgeschlossenen Knoten wird das Meßleitungsrelais gesetzt und die Relais des ersten Ausgangs werden nacheinander ein- und ausgeschaltet. Ist dies erfolgt, so wurden alle 10 Relais der beiden zusammengehörenden Ausgänge geprüft und der Test kann für die nächsten beiden Koppelpunkte beginnen.

Nun zur Schaltung der Selbstestplatine (Platine PL1):
Die Relais K1 bis K4 dienen dazu, die +24V auf die oberen vier Relais-Bus-Leitungen zu legen. K5 schaltet den Durchgangsprüfungseingang des Rechners auf die fünfte Relais-Bus-Leitung. Zum Schutz gegen Überspannungen auf den Ansteuerleitungen der Relais beim Ausschalten sind gegenpolig zur Versorgungsspannung jeweils Schutzdioden vom Typ 1N4001 an den Erregerspulen der Relais angeschlossen.


RELAIS-BUS
D0
3
4
5
6
D4
7
+24V
MESSPANNUNG
9
DURCHGANGSPR.
12
+24V
VERSORGUNG
31
D1
K1
D2
K2
D3
K3
D4
K4
D5
K5
ANSTEUERUNG
SELBSTTEST
D0
21
22
23
24
D4
25
SCHALTKONTAKTE
LEIST.PUNKT
15
16
D6
K6
ANSTEUERUNG
LEIST.PUNKT
D0
27
WOETRA B Selbsttestplatine (PL1)
GEZEICHNET : WOLFGANG SCHRAML
DATUM : 12/89
BLATT 1 VON 1

2.4.3. Meßgeräteumschaltung

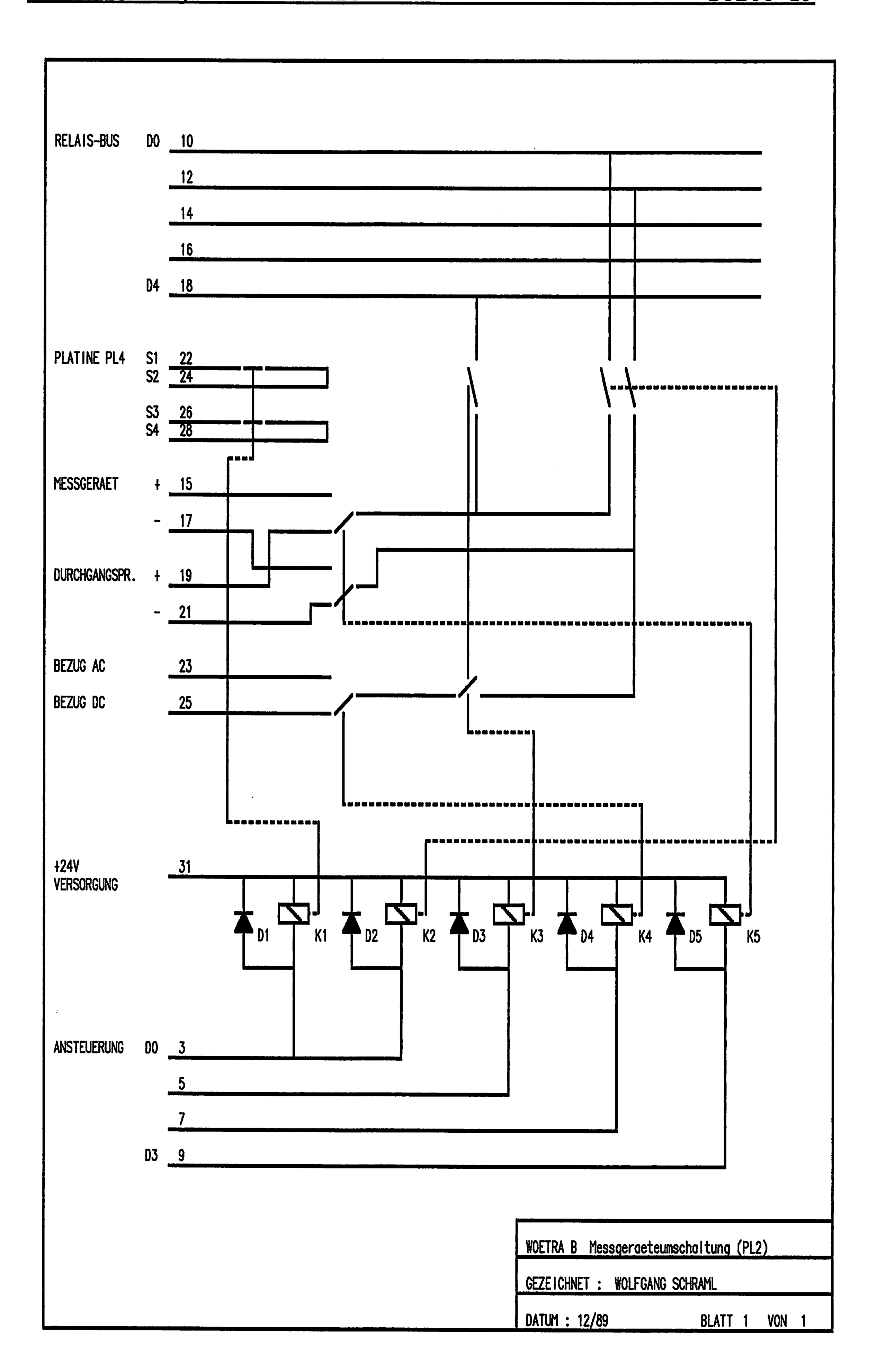
Die Meßgeräteumschaltung erfüllt die Aufgabe, zwischen Meßgerät und Durchgangsprüfungseingang umzuschalten, sowie den Bezugspunkt für die verschiedenen Meßarten festzulegen. Es können folgende Betriebszustände der Schaltung festgehalten werden:

- Durchgangsprüfung auf Relais-Bus-Leitung D4
  Relais K3 ist angezogen, alle anderen Relais befinden sich im Ruhezustand

- Meßgerät auf Relais-Bus-Leitung D4, Bezugspunkt interne Prüfspannungsmasse
  Die Relais K3 und K5 sind angezogen, alle anderen im Ruhezustand

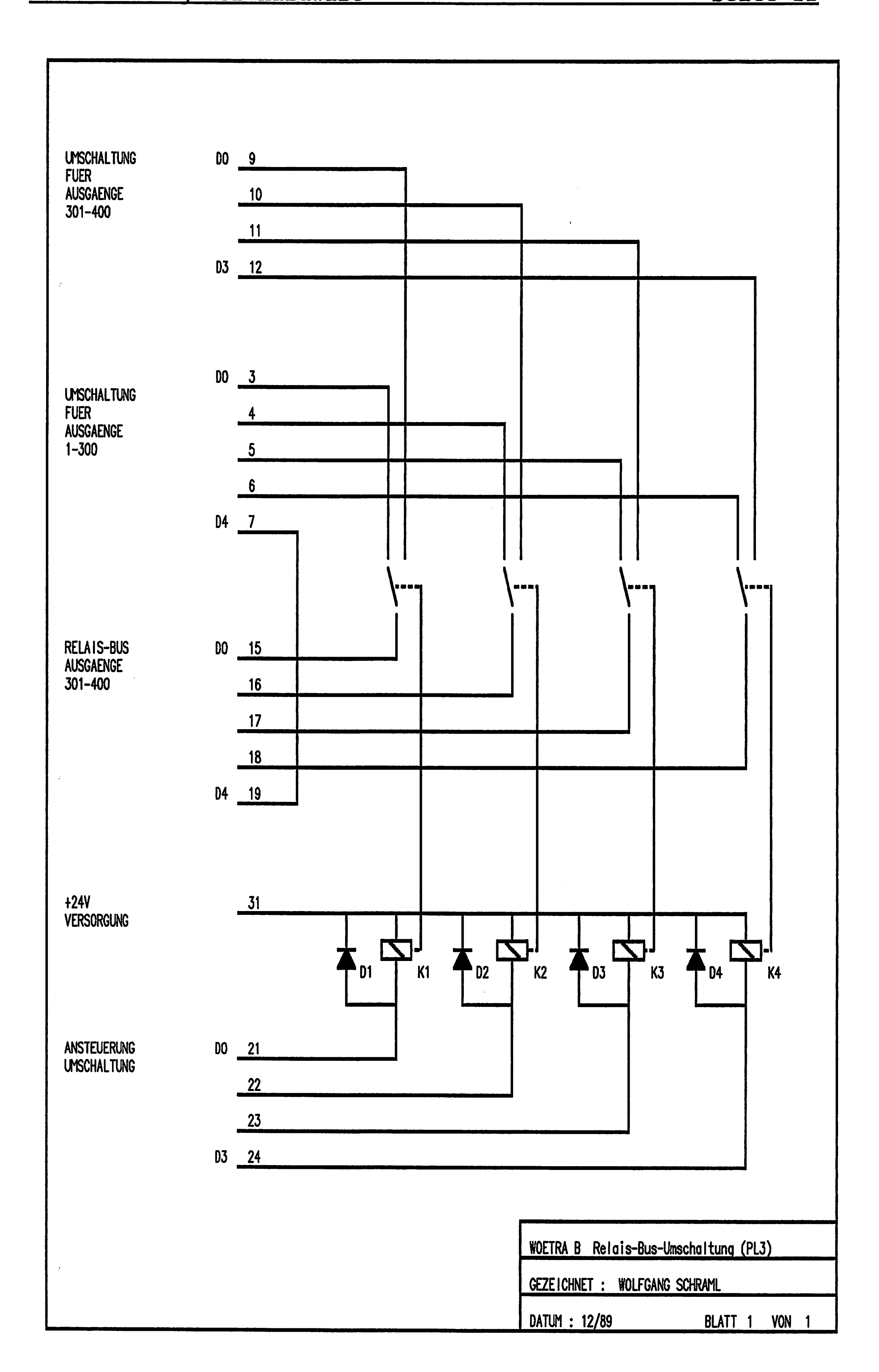
- Meßgerät auf Relais-Bus-Leitung D4, Bezugspunkt ist Nullpunkt der Prüf-Wechselspannungen
  Die Relais K3, K4 und K5 sind angezogen, der Rest im Ruhezustand

- Meßgerät auf den oberen beiden Relais-Bus-Leitungen (D0,D1)
  K1, K2 und K5 sind aktiv


RELAIS-BUS D0 10
12
14
16
D4 18
PLATINE PL4 S1 22
S2 24
S3 26
S4 28
MESSGERAET + 15
- 17
DURCHGANGSPR. + 19
- 21
BEZUG AC 23
BEZUG DC 25
+24V
VERSORGUNG 31
D1 K1 D2 K2 D3 K3 D4 K4 D5 K5
ANSTEUERUNG D0 3
5
7
D3 9
WOETRA B Messgeraeteumschaltung (PL2)
GEZEICHNET : WOLFGANG SCHRAML
DATUM : 12/89 BLATT 1 VON 1

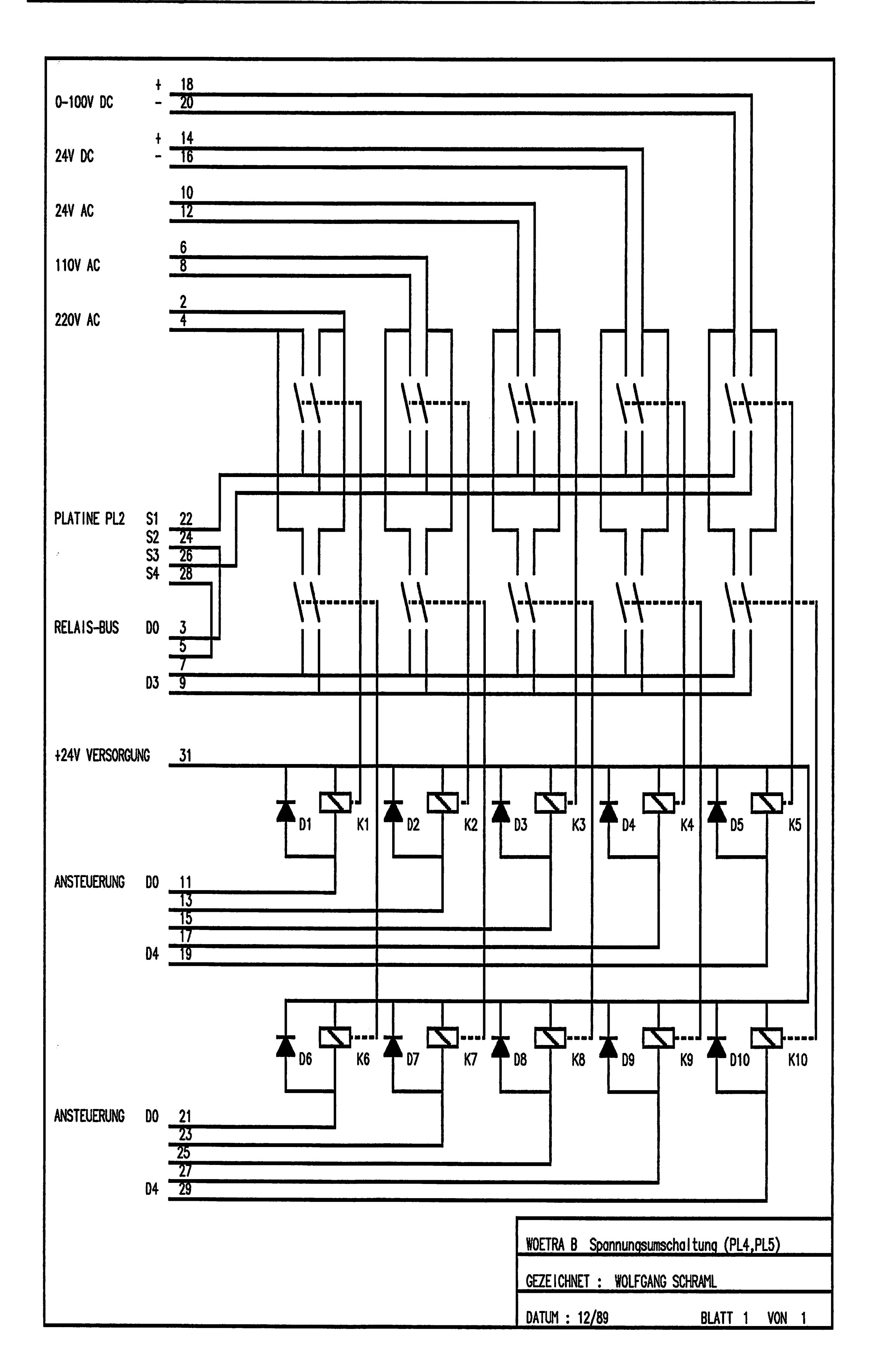
2.4.4. Relais-Bus-Umschaltung

Bei der jetztigen Version des WÖTRA sind lediglich fünf Relais-Bus-Leitungen implementiert. Da für jede anzulegende Spannung bzw. das Meßgerät zwei Leitungen benötigt werden, ist dies eine erhebliche Einschränkung der universellen Verwendung. Um dies wieder teilweise auszugleichen, wurde eine Umschaltung des Relais-Bus ab Ausgung Nr.301 vorgesehen. Dadurch hat man dann die Möglichkeit, bei umfangreicheren Prüfschritten den Relais-Bus aufzutrennen und auf den oberen 100 Ausgängen noch zwei zusätzliche Spannungen anzulegen.
Realisiert ist dies, indem die Relais-Bus-Leitungen bei Ausgang 300 auf eine Platine mit vier Umschaltrelais geführt sind. Im Ruhezustand der Relais sind die Leitungen mit denen der oberen 100 Ausgänge durchverbunden. Erst bei Ansteuerung aller vier Relais wird der Relais-Bus aufgetrennt und die Ausgänge 301-400 können nun unabhängig von den Ausgängen 1-300 beschaltet werden.


UMSCHALTUNG
FUER
AUSGAENGE
301-400
D0 9
10
11
D3 12
UMSCHALTUNG
FUER
AUSGAENGE
1-300
D0 3
4
5
6
D4 7
RELAIS-BUS
AUSGAENGE
301-400
D0 15
16
17
18
D4 19
+24V
VERSORGUNG
31
D1
K1
D2
K2
D3
K3
D4
K4
ANSTEUERUNG
UMSCHALTUNG
D0 21
22
23
D3 24
WOETRA B Relais-Bus-Umschaltung (PL3)
GEZEICHNET : WOLFGANG SCHRAML
DATUM : 12/89
BLATT 1 VON 1

2.4.5. Spannungsumschaltung

Die Spannungsumschaltung hat die Aufgabe, die Prüfspannungen wahlfrei auf die oberen vier Relais-Bus-Leitungen schalten zu können. Je fünf zweipolige Schaltrelais für je zwei Relais-Bus-Leitungen wurden hierfür eingesetzt. Je nachdem, welches der fünf Relais eingeschaltet wird, liegt eine der fünf Prüfspannungen auf den oberen zwei Relais-Bus-Leitungen oder den Leitungen D3 und D4. Die unterste Relais-Bus-Leitung wurde nicht mit einbezogen, da sie nur für Durchgangsprüfungen verwendet wird.
Diese Spannungsumschaltung wird beim WÖTRA B zweimal benötigt (PL4, PL5), da für den Fall des umgeschalteten Relais-Bus auch auf die Ausgänge 301-400 beliebige Prüfspannungen schaltbar sein sollen.
Außer den 10 Relais, welche die Umschaltung erledigen, enthält die Schaltung nur noch 10 Dioden zum Schutz vor Überspannungsspitzen beim Ausschalten.


0-100V DC
+ 18
- 20
24V DC
+ 14
- 16
24V AC
10
12
110V AC
6
8
220V AC
2
4
PLATINE PL2
S1 22
S2 24
S3 26
S4 28
RELAIS-BUS
D0 3
5
7
D3 9
+24V VERSORGUNG 31
D1 K1 D2 K2 D3 K3 D4 K4 D5 K5
ANSTEUERUNG D0 11
13
15
17
D4 19
D6 K6 D7 K7 D8 K8 D9 K9 D10 K10
ANSTEUERUNG D0 21
23
25
27
D4 29
WOETRA B Spannungsumschaltung (PL4,PL5)
GEZEICHNET : WOLFGANG SCHRAML
DATUM : 12/89
BLATT 1 VON 1

2.4.6. Leistungspunkt

Der Leistungspunkt ist ein unabhängig von den Prüfspannungen schaltbarer 220V Ausgang. Er ist dann notwendig, wenn Baugruppen geprüft werden sollen, die Netztrafos hoher Leistung beinhalten. Die 220V Prüfspannung, die auf den Relais-Bus geschaltet werden kann, ist in diesem Fall nicht geeignet, da sie nur für geringere Leistungen ausgelegt ist.
Um diesen Leistungsausgang schalten zu können, befindet sich auf der Platine PL1 noch zusätzlich ein Relais (K6), welches einen 220V-Schütz ansteuert. Dieser wiederum schaltet den Leistungspunkt am WÖTRA.

2.5. Adreßbelegung

Um die verschiedenen Umschaltplatinen ansprechen zu können, wurde zu den schon vorhandenen 25 Dekoderplatinen des Typs DW3 eine Weitere hinzugenommen. Die Basisadresse wurde auf den Wert 400 gelegt, und damit ergibt sich dann folgende Adreßbelegung der Umschaltungen:

400 --------------- Selbsttest
401 --------------- Meßgeräteumschaltung
402 --------------- Relais-Bus-Umschaltung
403 --------------- Spannungsumschaltung Ausgang 1-300
404 --------------- " "
405 --------------- Spannungsumschaltung Ausgang 301-400
406 --------------- " "
407 --------------- Leistungspunkt

2.6. Aufbau und Inbetriebnahme

Der Aufbau des Baugruppenprüfgeräts WÖTRA B kann in mehrere Phasen gegliedert werden. Zu Beginn des Projekts war es notwendig, erst einmal die gesamten mechanischen Arbeiten

durchzuführen. Dazu gehörte vor allem die Bearbeitung des Stahlschranks, in dem alle Komponenten des Geräts plaziert werden. Es handelt sich um einen Schrank der Firma Schroff, welcher bereits für den Einbau von 19 Zoll-Baugruppenträgern ausgelegt ist. Im Innern des Schranks wurden eine Reihe von Kabelkanälen eingebaut, welche dazu dienen, den 100 Verbindungskabeln zwischen der Dekodierung und den Relaiskarten die notwendige mechanische Stabilität zu verleihen. Dies ist deshalb wichtig, damit die Steckverbinder auf beiden Seiten der Kabel durch ihr schon erhebliches Eigengewicht nicht zu sehr belastet werden.
Ein weiterer großer Block unter den mechanischen Arbeiten war die Herstellung der Ausbrüche an der Frontplatte des Geräts (u.a. für das regelbare Netzgerät, die verschiedenen Anzeigeinstrumente für die Prüf- und Betriebsspannungen, die Einbau-Sicherungsautomaten und die 39pol VK-Buchsen für die Adaptierung der Prüflinge).
Ferner mußten noch die zehn 19-Zoll-Baugruppenträger zur Aufnahme der 100 Relaiskarten und die zwei Doppelrahmen für die 26 Dekoderplatinen vorbereitet werden (Zusammenbau der Rahmenteile und Montage der Steckverbinder an der Rückseite, Einbau der Busplatinen).
Nachdem all diese Vorarbeiten erledigt waren, konnte mit dem elektrischen Aufbau begonnen werden.
Der Hauptteil lag in der Anfertigung der 100 Verbindungskabel zwischen Dekoder- und Relaisplatinen und dem Herstellen der Wrap-Verbindungen (Durchschleifen des Relaisbus über alle 100 Karten, Verbinden der 400 Knotenpunkte mit den Steckverbindern an der Frontplatte) auf der Rückseite der Baugruppenträger für die Relaiskarten. Die Wire-Wrap-Technik wurde deshalb eingesetzt, weil sie sehr gute Kontakt- und Haltbarkeitseigenschaften aufweist und auch relativ schnell ausführbar ist.
Nachdem die genannten Vorarbeiten erledigt waren (überwiegend von Ferienarbeitern und Praktikanten unter meiner Anleitung und Aufsicht), kam mein Teil der

Aufbauarbeiten: die Endmontage des gesamten Prüfgeräts und dessen Inbetriebnahme.
Hierzu wurde wie folgt vorgegangen: um den Überblick nicht zu verlieren, wurden die Dekoder- und Relaisplatinen Schritt für Schritt in den Schrank eingebaut und erst ausgetestet, bevor mit dem Aufbau fortgefahren wurde. Um leicht überschaubare Einheiten zu erhalten, wurden jeweils eine Dekoder- und die vier zugehörigen Relaiskarten zusammen mit den hierfür benötigten 16 Verbindungskabeln montiert. War dies geschehen, so wurde mit Hilfe des Selbsttestprogramms, welches später noch ausführlich beschrieben wird, ein erster Test der bestückten Platinen vorgenommen. Verlief er fehlerfrei, so konnte mit dem Aufbau fortgefahren werden. Trat ein Fehler auf (und dies war aufgrund der Komlexität des Aufbaus mehrfach der Fall), so wurde er durch die entsprechende Meldung der Selbsttestroutine ("Relais Nummer xy, kein Kontakt bei Ein")erst einmal grob eingegrenzt. Wie bereits in der Beschreibung der Selbsttestplatine angesprochen, konnte dies mehrere Ursachen haben: fehlerhafte Ansteuerung, fehlerhafte Verbindungskabel zwischen Dekoder- und Relaiskarten, defektes Relais, durchgebrannte Sicherung auf der Relaiskarte, fehlerhafte Verbindungsleitung zu den Steckverbindern an der Frontplatte, Fehler in den Wrap-Verbindungen an den Aufnahmerahmen für die Relaiskarten. Die beiden letzten Fehlerarten wurden als Verursacher bereits dadurch ausgeschlossen, daß sämtliche Verbindungen auf dem Wrap-Rahmen sowie die Verbindungskabel zur Frontplatte vorher mit einem Durchgangsprüfer nachgemessen wurden. Die Verbindungskabel zwischen Dekoder-und Relaiskarten konnten auch als fehlerfrei betrachtet werden, da sie nach ihrer Fertigung durch Ferienarbeiterinnen mit einem kommerziellen Kabelbaumprüfgerät getestet und eventuell vorhandene Fehler sofort beseitigt wurden. Die Fehlersuche konzentrierte sich also auf die Dekodierung und die Relaiskarten. Zunächst wurden die Sicherungen auf den Platinen überprüft. Hier konnten bereits viele Fehler durch Auswechseln dieser behoben werden. Führte dies nicht zum

Erfolg, so überprüfte ich die Verbindungen zwischen dem Ausgang der Dekodierung und dem Eingang der entsprechenden Relaiskarte. Hier traten mehrere Fehler dadurch auf, daß beim Stecken der Kabel einige der sehr empfindlichen Wrap-Stachel verbogen wurden und deshalb keinen Kontakt mehr hatten. Auch diese Fehlerquelle konnte durch das Zurechtbiegen der betroffenen Kontakte leicht behoben werden. Einige sehr hartnäckige Defekte waren jedoch auch an dieser Stelle noch nicht zu lokalisieren. Deshalb wurden anschließend die für den fehlerhaften Ausgang zuständigen Optokoppler getauscht. Dies führte dann zum gewünschten Erfolg. Nur in einem Falle war dies nicht so. Es war die gesamte Dekoderplatine "tot", d.h. sie zeigte keinerlei Funktion. Woran dies lag, konnte wegen des Zeitdrucks während des Aufbaus nicht ermittelt werden. Die defekte Platine wurde gegen eine ordnungsgemäß funktionierende ausgetauscht, wodurch dann keine Probleme mehr auftauchten. Waren dann schließlich auf diese Weise alle Fehler einer bestückten Zeile behoben, so konnte der nächste Aufbauschritt in gleicher Weise durchgeführt werden.
Nach Beendigung der Montagearbeiten erfolgte dann noch ein abschließender Selbsttest des gesamten, komplett bestückten Geräts. Auf diese Weise konnte zumindest davon ausgegangen werden, daß das Gerät im Prinzip funktionstüchtig war und später auftretende Fehler in der Software zu Suchen waren.

2.7. Spannungsüberwachung

Sämtliche Betriebs- und Prüfspannungen des WÖTRA B können über Anzeigeinstrumente an der Frontplatte überwacht werden. Es sollte jedoch zusätzlich noch die Möglichkeit bestehen, die Spannungen vom Rechner kontrollieren zu lassen. Dies ist deshalb sinnvoll, da der Bediener des WÖTRA im Normalfall seine ganze Aufmerksamkeit der gerade zu prüfenden Baugruppe schenkt und nicht andauernd die Anzeigen auf der Frontplatte überwacht. Es sollte daher in bestimmten Zeitabständen eine Kontrolle der Spannungen durch den Rechner erfolgen.

Hardwaremäßig soll die Realisation so erfolgen, daß die Betriebs- und Versorgungsspannungen über Isolierverstärker an die analogen Eingänge der universellen Portkarte geführt werden. Meine Aufgabe war es dabei, eine Versuchschaltung für die galvanisch getrennte Übertragung eines Spannungskanals aufzubauen.

Grundlage dieser Schaltung ist der Baustein 1B21, ein zwischen Ein- und Ausgang galvanisch getrennter U/I-Wandler. Die Eingangsbeschaltung wurde laut Datenblatt so vorgenommen, daß einem Eingangsspannungsbereich von 0-5V ein Ausgangsstrombereich von 0-20mA entspricht. Als Versorgungsspannung für den Ausgangskreis (U1) wurde ein 15V-Netzteil verwendet, wodurch sich ein maximaler Lastwiderstand am Ausgang von 750 Ohm ergibt (15V/20mA).

Der Ausgangsstrom von 0-20mA wird durch einen I/U-Wandler wieder in einen Spannungsbereich von 0-5V übersetzt. Hierfür fand eine Standard-OPV-Schaltung Verwendung. Der Rückkopplungswiderstand R errechnet sich aus der bei 20mA zu erhaltenden Spannung von 5V zu 250 Ohm. Um einen genauen Abgleich vornehmen zu können, wurde dann beim Aufbau der Schaltung ein 500 Ohm Potentiometer eingesetzt. Die Versorgungsspannung für den OPV wurde einem externen Netzteil entnommen.

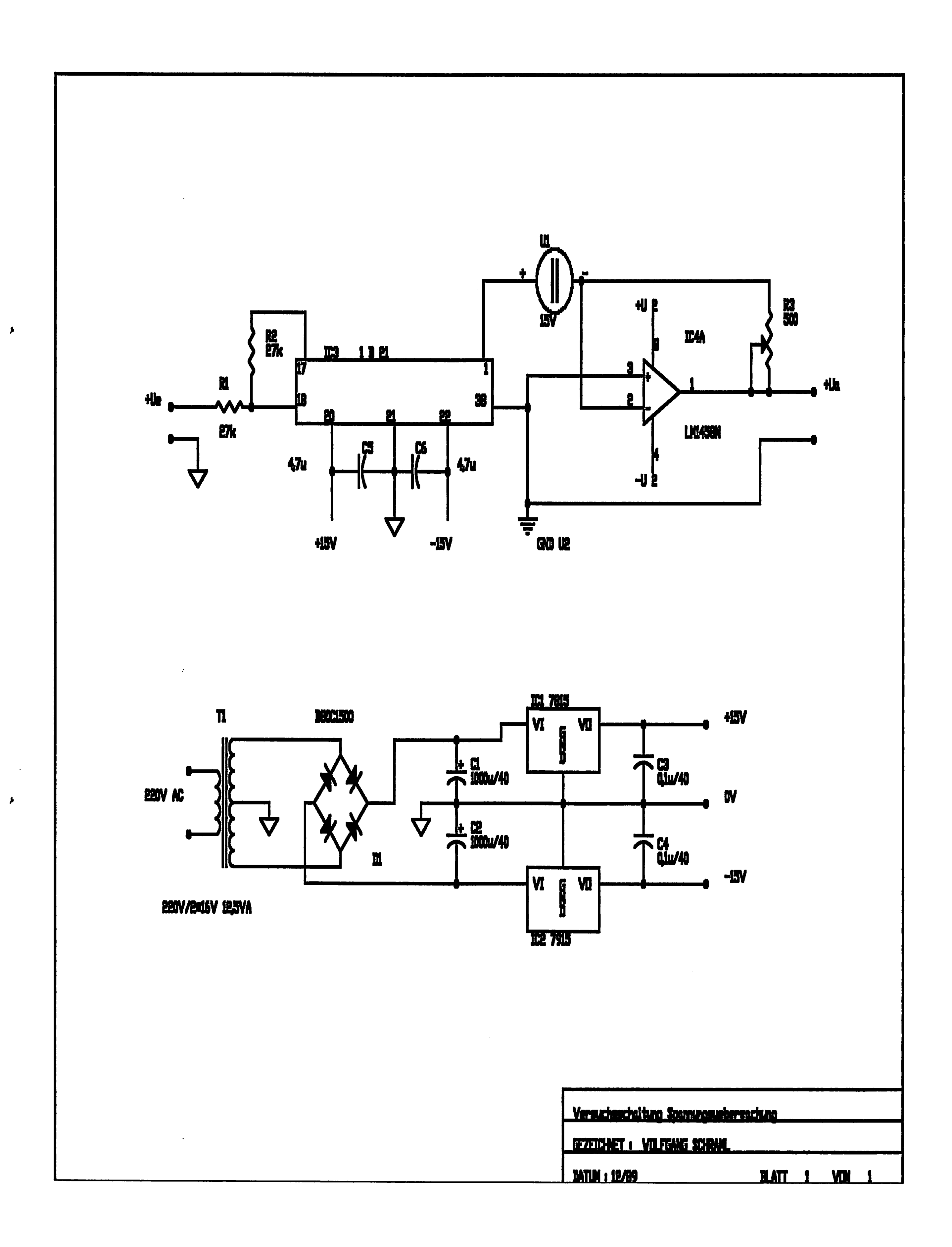
Da es sich um eine Versuchsschaltung handelt, wurde sie auf einer Lochrasterplatine im Einfach-Europakartenformat aufgebaut. Die Verdrahtung der Bauteile erfolgte mit lötbarem Fädeldraht.

Nach dem Aufbau wurde die Platine an die erforderlichen Spannungsversorgungen angeschlossen und getestet. Dazu wurde die Eingangsspannung auf 5V gestellt und die Ausgangsspannung mit dem Poti ebenfalls auf diesen Wert abgeglichen. Wurde dann der Eingangsspannungsbereich von 0 bis 5V durchfahren, konnte dieselbe Spannung auch am Ausgang gemessen werden. Damit war die korrekte Funktion der Schaltung sichergestellt.

Um die Schaltung später auch im WÖTRA einsetzen zu können, muß sie auf der Eingangsseite noch um einen Spannungsteiler erweitert werden, der die zu überwachende Spannung auf den

Wert von 5V begrenzt. Die Umrechnung auf den tatsächlich anliegenden Wert muß dann vom Programm, daß die Überwachung durchführt, getätigt werden.
Der Aufbau der gesamten Überwachungsschaltung und die Entwicklung des zugehörigen Programms waren nicht mehr Teil meiner Arbeit. Im später beschriebenen Menüprogramm wurde jedoch bereits der entsprechende Auswahlpunkt vorgesehen, so daß eine Erweiterung ohne Probleme möglich ist.

Es wäre noch zu überlegen, ob statt des Bausteins 1B21 ein Isolationsverstärker verwendet werden sollte, der am Ausgang wieder direkt eine Spannung liefert. Dadurch könnte für jeden zu übertragenden Kanal die Spannungsquelle im Ausgangszweig eingespart werden. Als geeigneter Baustein wäre z.B. der Isolationsverstärker ISO 120 in Verbindung mit dem DC-DC-Wandler PWS 740 von Burr-Brown zu nennen.
Welche der beiden Alternativen letztendlich Verwendung finden wird, ist zum momentanen Zeitpunkt noch nicht vollständig geklärt. Es wird jedoch mit ziemlicher Sicherheit aus Kostengründen für das jetzige Gerät der 1B21 zum Einsatz kommen (er wurde bereits in der benötigten Stückzahl vor Beginn meiner Diplomarbeit bestellt). Für die in Zukunft zu bauenden Geräte sollte zur Vereinfachung der Schaltungen jedoch das andere System in Betracht gezogen werden.


U1
15V
+U
IC4A
R3
500
R2
27k
IC3
R1
+Ue
27k
+Ua
LM1458N
-U
C5
C6
4,7u
4,7u
+15V
-15V
GND U2
T1
B80C1500
IC1 7815
+15V
VI
VO
GND
C1
1000u/40
C3
0,1u/40
220V AC
0V
C2
1000u/40
C4
0,1u/40
D1
-15V
220V/2x15V 12,5VA
IC2 7915
DATUM : 12/89
BLATT 1 VON 1

3. Beschreibung der Software

3.1. Allgemeines

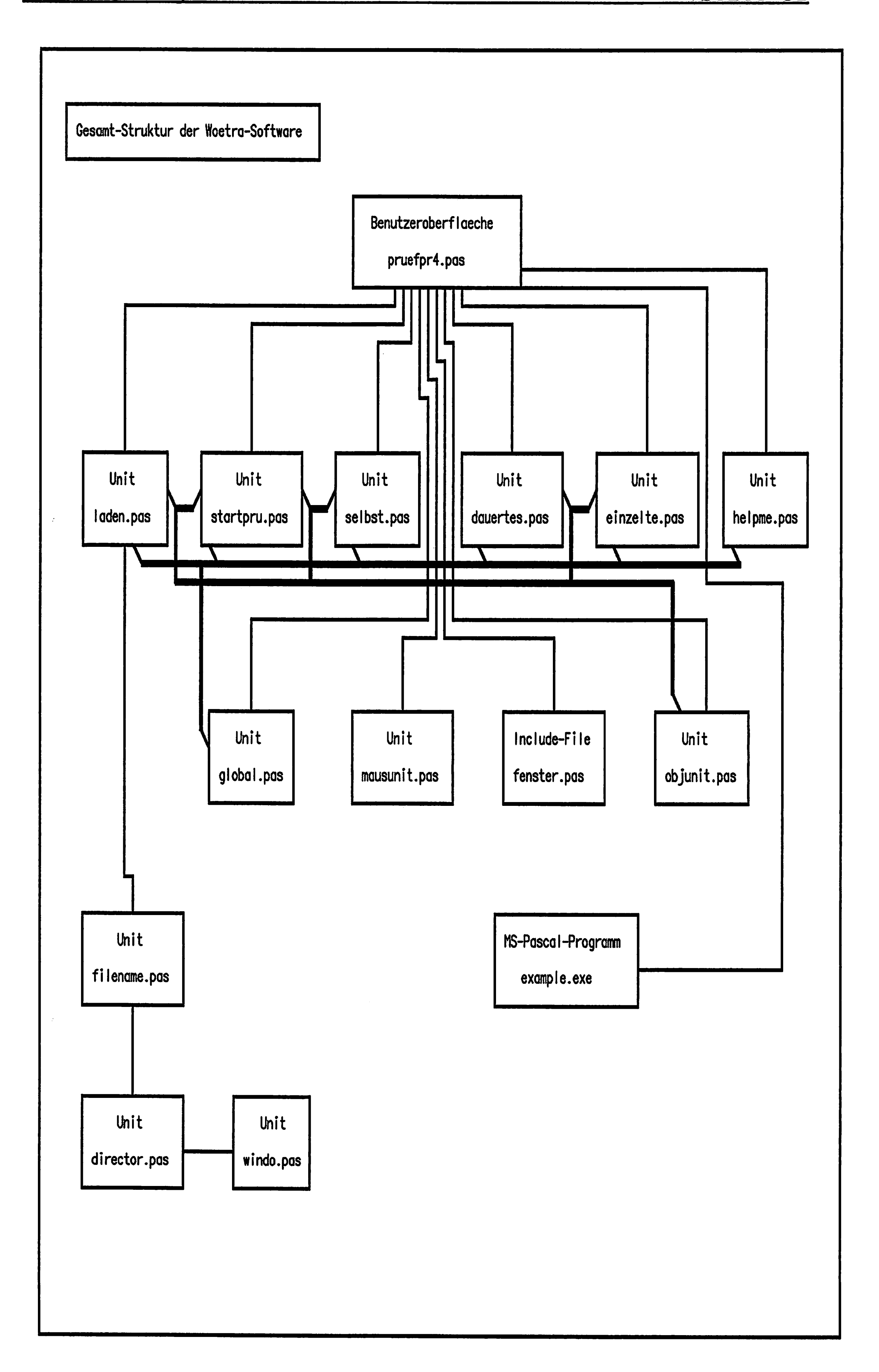
Zu Beginn der Softwareerstellung wurde folgender Pflichtenkatalog aufgestellt:

- sämtliche Funktionen und Programme zur WÖTRA-Steuerung sollten von einem übergeordneten Menüprogramm aus erreichbar sein.
- die Bedienung der Programme hat möglichst einfach zu sein
- Fehleingaben sollten so weit als möglich bereits vom Programm abgefangen werden, d.h. ein unkontrollierter Ausstieg aus den Programmen darf nach Möglichkeit nicht vorkommen
- die Bildschirmgestaltung in den einzelnen Unterprogrammen sollte immer gleich sein, um eine Verwirrung des Bedieners von vornherein auszuschließen
- die Farbgebung der Programme sollte nicht zu überladen wirken, um das Arbeiten am Rechner für die Augen nicht noch anstrengender zu machen, als es ohnehin schon ist
- auf der Ebene des Menüprogramms sollten möglichst zu jedem Punkt "Online" Hilfefunktionen vorhanden sein, um es dem Prüfer zu ermöglichen, wichtige Punkte schnell nachlesen zu können, ohne lang in Bedienungsanleitungen zu suchen.

Wie die einzelnen Forderungen dann in die Praxis umgesetzt wurden, ist aus der nachfolgenden Detailbeschreibung der Software zu ersehen.

3.2. Zusammenwirken der Programmteile

siehe nächste Seite


Gesamt-Struktur der Woetra-Software
Benutzeroberflaeche
pruefpr4.pas
Unit
laden.pas
Unit
startpru.pas
Unit
selbst.pas
Unit
dauertes.pas
Unit
einzelte.pas
Unit
helpme.pas
Unit
global.pas
Unit
mausunit.pas
Include-File
fenster.pas
Unit
objunit.pas
Unit
filename.pas
MS-Pascal-Programm
example.exe
Unit
director.pas
Unit
windo.pas

3.3. Global verwendete Prozeduren und Funktionen

In diesem Kapitel sollen all jene Prozeduren beschrieben werden, die grundlegende Ein- Ausgabefunktionen bereitstellen und in den einzelnen Programmen und Units immer wieder verwendet wurden. Hierzu gehören im Einzelnen:

- Die Prozeduren zur Maussteuerung
- Die Prozeduren und Funktionen zur Datenein- und ausgabe über die universelle Portkarte
- Prozeduren zur einfacheren Bildschirmsteuerung
- das Einlesen von Meßwerten über die IEC-Bus Schnittstelle vom verwendeten Multimeter Fluke 8842A
- die über <F1> erreichbaren Hilfe-Funktionen

3.3.1. Das Unit mausunit.pas

In diesem Unit sind alle Unterprogramme zur Maussteuerung zusammengefaßt. Sie wurden alle in enger Anlehnung an (1*) für die hier benötigten Zwecke übernommen.
Sämtliche Mausfunktionen werden über den DOS-Interrupt 33h erreicht, welcher durch die Einbindung eines Maustreibers in die PC-Umgebung entsprechend installiert wird.
Zu Beginn des Units werden alle Unterprogramme, die von außen zugänglich sein sollen, mit ihrem Definitions- und Variablenkopf aufgeführt. Anschließend folgt im Implementationsteil die eigentlichen Prozedur-Quelltexte. Dies sind im Einzelnen:

3.3.1.1. Prozedur Maus

Diese Prozedur wird von allen anderen in dieser Unit vorhandenen Prozeduren dazu benutzt, um den DOS-Interrupt 33h aufzurufen. Es werden am Anfang die benötigten Register mit den je nach gewünschter Mausfunktion verschiedenen Wer-

(1*) siehe Literaturverzeichnis unter MaNager

ten geladen. Dann wird der Mausinterrupt aufgerufen und die entsprechenden Registerwerte wieder an die verwendeten Variablen zurückgegeben.

3.3.1.2. Prozedur Zeige_Maus

Sie dient dazu, um den Mauscursor am Bildschirm sichtbar zu machen. Die benötigte Funktionsnummer des Interrupts ist 1. Dieser Wert wird an das AX-Register übergeben und die Prozedur Maus aufgerufen. Es werden keine Werte vom Interrupt zurückgeliefert.

3.3.1.3. Prozedur Verstecke_Maus

Sie bewirkt genau das Gegenteil von "Zeige_Maus" und läßt den Mauscursor wieder vom Bildschirm verschwinden. Die Funktionsnummer im AX-Register ist 2, sonst ist der Ablauf wie unter "Zeige_Maus" beschrieben.

3.3.1.4. Prozedur Maus_Position

Ermittelt die Koordinaten des Mauscursors am Bildschirm. Sie werden in den Variablen x und y vom Datentyp "Word" an das aufrufende Programm zurückgeliefert. Die Funktionsnummer im AX-Register ist 3, die Koordinaten werden nach dem Aufruf des Mausinterrups in den Registern CX und DX zurückgegeben. Hier ist folgendes zu beachten: während der normale Textcursor x-Koordinaten im Bereich von 0-79 und y-Koordinaten im Bereich von 0-24 aufweist, werden die Mauskoordinaten in den Bereichen 0-639 und 0-349 vergeben (entsprechend der Auflösung der EGA-Karte). Dies bedeutet, daß das Vorrücken des Textcursors um eine Stelle dem Vorrücken des Mauscursors um 8 Stellen entspricht !

3.3.1.5. Prozedur Setze_Maus

Bewegt den Mauscursor an eine beliebige Stelle des aktuellen Bildschirms, welche durch die zu übergebenden Variablen

x und y spezifiziert wird. x muß wieder im Bereich von 0-639, y im Bereich von 0-349 liegen. Die Funktionsnummer für diese Routine ist 4.

3.3.1.6. Prozedur Reset_Maus

Bringt sämtliche Mausparameter wieder in den ursprünglichen Zustand. Die Interrup-Funktionsnummer für diese Betriebsart ist Null. Sie muß im AX-Register stehen.

3.3.1.7. Funktion Knopf_Links

Ermittelt den Status der linken Maustaste. Liefert den Wert "TRUE" zurück, wenn die Taste gedrückt ist, "FALSE", wenn sie nicht gedrückt ist. Hierzu wird mit der Funktion "Bittest" der Wert im BX-Register überprüft, der je nach Status der Taste 0 oder 1 ist.

3.3.1.8. Funktion Knopf_Rechts

Wie "Knopf_Links", nur für die rechte Maustaste.

3.3.1.9. Funktion Maus_Im_Bereich

Überprüft, ob sich der Mauscursor in dem durch die Variablen x1,y1,x2,y2 festgelegten Bildschirmausschnitt befindet. Ist dies der Fall, so hat die Funktion den Wert "TRUE", andernfalls "FALSE".

3.3.1.10. Prozedur Aendere_Verhältnis

Bestimmt das Verhältnis von physikalischer Mausbewegung zur Mausbewegung am Bildschirm. Ist dieses Verhältnis groß, so wirkt sich eine Bewegung der Maus am Bildschirm nicht so stark aus, die Mausbewegung erscheint langsamer.

3.3.2. Das Unit global.pas

In diesem Unit sind alle Unterprogramme zusammengefaßt, die von allen anderen Programmteilen aus zugänglich sein sollen. Es enthält auch die Vereinbarung von globalen Datentypen. Hierbei ist als Wichtigster der Datentyp "einzelschritt" zu nennen, der das Record für den Aufbau der Prüfdateien definiert. Da seine genaue Beschreibung im Kapitel über den Menüpunkt "Start" erfolgt, soll an dieser Stelle darauf verzichtet werden.

3.3.2.1. Die Prozedur Ausgeben

Sie kann als Kernstück der gesamten WÖTRA-Software betrachtet werden. Mit ihrer Hilfe ist es möglich, auf einer bestimmten WÖTRA-Adresse (bzw. Ausgang) die gewünschten Daten, d.h. die Relais, die geschaltet werden sollen, auszugeben. Diese Prozedur wird vom Prüfablaufprogramm und von sämtlichen Testroutinen sowie von der Reset-Prozedur benötigt (3).
Die Ausgabe erfolgt über die Ports a,b und c des Bausteins 8255, der sich auf der universellen Portkarte im Rechner befindet (4). Zu Beginn der Prozedur werden den drei Variablen "aport", "bport" und "cport" Low- und High-Byte der auszugebenden Adresse sowie die gewünschten Daten übergeben. Anschließend erfolgt die Ausgabe dieser Werte unter Zuhilfenahme des PORT-Befehls von Turbo-Pascal an die Schnittstellenkarte. Die FOR-Schleifen, die zwischen den Ausgabebefehlen eingebaut sind, dienen dazu, eine gewisse Verzögerung einzubauen, um den Optokopplern genügend Zeit zum Durchschalten zu lassen. Liegen alle Adreß- und Datenbits am Ausgang an, so wird durch ein kurzes Setzen des Write-Signals auf Low der Dekodierelektronik signalisiert, daß die Logikpegel in diesem Moment gültig sind und

---

(3) siehe auch Diplomarbeit Stefan Seitz

(4) siehe Handbuch Portkarte me-30

übernommen werden können. Ist das geschehen, so geht das Write-Signal wieder in den High-Zustand und die Ausgabe ist beendet.
Da pro Knoten immer nur ein Relais geschaltet werden darf, sind folgende Datenwerte für die Ausgabe zulässig:
1 für das Schalten des ersten Relais eines Knotens, 2 für das Schalten des zweiten, 4 für das Schalten des dritten bis 16 für das Schalten des fünften.

3.3.2.2. Prozedur Port_Programmieren

Der 8255 ist als universeller Ein-Ausgabe Baustein konzipiert worden. Da er im Falle des WÖTRA B nur zur Ausgabe dient, muß er mit Hilfe des internen Steuerregisters in diesen Modus gebracht werden. Das entsprechende Steuerbyte ist 80h, es wird durch die Prozedur "Port_Programmieren" in das Steuerregister des 8255 geschrieben.

3.3.2.3. Prozedur Adport

Die universelle Portkarte, die hier Verwendung findet, besitzt außer den 24 digitalen Ausgängen noch zusätzlich 16 analoge Eingänge. Auch diese werden über einen Portbaustein des Typs 8255 geführt. Um eine korrekte Funktion zu erhalten, muß dieser in den Modus "Eingabe" geschaltet werden. Dies bewerkstelligt die Prozedur "Adport". Sie schreibt das Steuerbyte 92h in das entsprechende Register des 8255.

3.3.2.4. Funktion Spannung_Lesen

Sie dient zum Einlesen eines analogen Spannungswerts über die Portkarte des PCs. Als Variable wird die Nummer des gewünschten A/D-Kanals (einer von 16) an die Funktion übergeben. Der Quelltext wurde direkt aus dem Handbuch zur Erweiterungskarte der Firma Impec übernommen.

3.3.2.5. Prozedur Relais_Reset

Beim Einschalten des WÖTRA sind die Flip-Flops auf den Dekodierplatinen alle gesetzt, d.h. alle Relais des WÖTRA ziehen an. Da dies zu einer Reihe von Kurzschlüssen auf den Relais-BusLeitungen führt, muß vor jedem Aufruf eines Test- oder Prüfprogramms ein Reset durchgeführt werden, der alle Relais wieder in den erregungslosen Zustand versetzt. Der Reset wird verwirklicht, indem eine Schleife alle Adressen von 0 bis 400 durchläuft und jedesmal eine Null als Daten ausgegeben wird.

3.3.2.6. Prozedur Umschaltung_Reset

Die Prüfung einer Baugruppe erfolgt aufgrund der Verwendung einer Prüfdatei immer Schritt für Schritt, wobei jedesmal eine neue Konfiguration der Umschaltplatinen erforderlich ist. Damit es hier nicht zu Überschneidungen kommen kann, welche Kurzschlüsse zur Folge hätten, wird nach jedem Prüfschritt ein Reset der Umschaltplatinen durchgeführt, d.h. die Adressen 400 bis 410 werden in einer Schleife durchlaufen und durch Ausgabe einer Null zurückgesetzt.

3.3.2.7. Prozedur Cursor_Off

Bei der Erstellung einer Bildschirmmaske ist es oft notwendig, den Cursor unsichtbar zu machen, da er sich nur störend auswirken würde. Dies kann erreicht werden, indem man den Cursor durch Aufruf des BIOS-Interrups 10h einfach aus dem aktuellen Bildschirmbereich entfernt. Dazu wird das AX-Register mit der entsprechenden Funktionsnummer des Interrupts geladen. Im BX- und DX-Register stehen dann die Parameter, die den Cursor verschwinden lassen.

3.3.2.8. Prozedur Knoten_Lesen

Bei den verschiedenen Prüf- und Testprogrammen des WÖTRA muß der Benutzer zu Beginn immer eingeben, welche Ausgänge

oder Relais getestet bzw. ab welchem Prüfschritt die Prüfung erfolgen soll. Alle diese Eingaben müssen innerhalb einer Bildschirmmaske getätigt werden. Drückt nun der Bediener versehentlich die <RETURN>-Taste, ohne vorher einen Wert eingegeben zu haben, führt dies zwar programmtechnisch nicht zu einem Fehler, die Bildschirmmaske wird jedoch durch das unvorhergesehene Weiterrücken des Cursors um eine Bildschirmzeile zerstört. Da dies keine zufriedenstellende Lösung sein konnte, wurde überlegt, wie dieses Drücken der <RETURN>-Taste abzufangen wäre. Das Ergebnis dieser Überlegungen ist die Prozedur "Knoten_Lesen".
Am Anfang wird der lokalen Variablen "zahlenfolge" der Leerstring zugewiesen. Dann bleibt das Programm solange in einer Schleife stehen, bis eine Ziffer von 0-9 eingegeben wird. Der Vorteil dieses Vorgehens liegt darin, daß falsch eingegebene Zeichen (und damit auch <RETURN>) schon bei der Eingabe unberücksichtigt bleiben. Ist nun eine gültige Ziffer eingegeben worden, wird sie in der Variablen "zahlenfolge" abgespeichert und durch eine Write-Anweisung am Bildschirm als Echo ausgegeben. Das ist notwendig, weil die Turbo-Pascal Funktion "ReadKey", welche hier verwendet wird, selbst kein Echo am Bildschirm erzeugt.
Nun wird in einer weiteren Schleife ähnlich der ersten der Rest der eingegebenen Ziffern ausgewertet. Dies erfolgt solange, bis entweder <RETURN> als Zeichen für das Ende der Eingaben gedrückt wird, oder maximal drei Stellen eingegeben wurden. Drei Stellen deshalb, weil die Adressen maximal dreistellig sind (1 bis 400). Zum Schluß der Prozedur wird noch die in der Variablen "zahlenfolge" abgespeicherte, eingegebene Ziffernfolge durch Auswertung mit Hilfe der "Val"-Prozedur in einen INTEGER-Wert gewandelt, der dann vom Programm weiterverarbeitet werden kann.

3.3.3. Das Unit objunit.pas

Die gesamte WÖTRA-Software ist in Turbo-Pascal Version 5.5 geschrieben. Sie ist kompatibel zur Version 5.0, weist aber sogenannte objektorientierte Erweiterungen auf.
Da eine Diplomarbeit auch dazu dienen soll, Neues auszuprobieren und Erfahrugen damit zu sammeln, regte mein Betreuer an, sich einmal mit dieser Art der Programmierung auseinanderzusetzen. Um auch für den WÖTRA verwendbare Ergebnisse zu erzielen, wurde ein Unit in objektorientierter Programmierung erstellt, welches dazu dient, am Bildschirm beliebige Umrahmungen zu zeichnen. Doch zunächst eine kurze Einführung in die OOP (Objekt-Orientierte-Programmierung).
Bei der bisherigen Version von Turbo-Pascal, welche eine rein prozedurale Sprache war, erfolgte eine Strenge Trennung von Daten einerseits und Prozeduren und Funktionen andererseits. Damit Daten verarbeitet werden konnten, mußten sie über die Parameterliste an die Unterprogramme übergeben werden. In der OOP hingegen können Daten und Prozeduren einen neuen Datentyp bilden, die sogenannten Objekte. Hierbei stehen die Datenfelder der Objekte für deren Inhalt an Informationen, die Prozeduren oder auch Funktionen für die Methoden, deren sie mächtig sind. All dies kann jedoch am Besten an einem Beispiel gezeigt werden: des Units "objunit.pas".
Wie bereits gesagt, soll es dazu dienen, beliebige Fensterumrahmungen am Bildschirm zu zeichnen. Diese Aufgabenstellung legt schon fest, welche Art von Daten und welche Prozeduren benötigt werden. Das sind zum einen die vier Koordinatenpaare der gewünschten Eckpunkte der Rahmen, zum anderen drei Prozeduren zur Übergabe der Koordinaten an das Objekt, welches die Umrahmung durchführen soll, zum Zeichnen des Rahmens und zum Löschen des Rahmens. Bei der herkömmlichen Art der Programmierung hätte jede der Prozeduren eine Parameterliste benötigt, damit die Koordinaten hätten übergeben werden können. In unserem Fall wird jedoch statt dessen zu Beginn des Units ein neuer Datentyp definiert, ein Objekt. Von der Syntax ist die Definition eines Objekts

der eines Records oder Verbundes sehr ähnlich. Die Daten des Objekts sind die Koordinaten des linken oberen und rechten unteren Eckpunktes des gewünschten Rahmens, die Variablen x1,y1,x2,y2. Dann folgt die Prozedur "Init", die die Aufgabe hat, die Parameter einmal an das Objekt zu übergeben. Das Objekt merkt sich dann sozusagen diese Koordinaten und kann dann später seine Methoden darauf anwenden. Die Prozedur "Zeigen" zeichnet den Rahmen, die Prozedur "Loeschen" läßt ihn wieder vom Bildschirm verschwinden.
Beim Zeichnen des Rahmens wird so vorgegangen, daß zuerst die vier Eckpunkte mit Blockgrafikzeichen des erweiterten IBM-Zeichensatzes dargestellt werden. Dann wird noch durch zwei FOR-Schleifen oberer und unterer sowie linker und rechter Begrenzungsrand gezogen. Die Prozedur "Loeschen" ist identisch aufgebaut, nur daß hier anstatt der Blockgrafikzeichen Leerzeichen ausgegeben werden, was dann den Rahmen wieder verschwinden läßt.
Nun zu einem konkreten Beispiel. Angenommen, es soll um die Bildschirmmitte ein kleines Rechteck gezeichnet und wieder gelöscht werden. Nennen wir es einmal "fenstmitt". Es muß jetzt zuerst in der Type-Anweisung des Programms ein Objekt wie oben beschrieben definiert werden. Es erhalte den Namen "fenster". Danach kann "fenstmitt" im Variablenteil des entsprechenden Programms als Datentyp "fenster" deklariert werden. Durch diese Anweisung wird "fenstmitt" auch zu einem Objekt der Klasse "fenster", es hat also dieselben Eigenschaften und Methoden wie dieses zur Verfügung. Diesen Vorgang bezeichnet man als Vererbung. Mit dem Befehl "fenstmitt.Init(x1,y1,x2,y2)" wird dem Objekt mitgeteilt, welche Daten es zur Verfügung haben soll. Ist dies geschehen, so kann der Rahmen "fenstmitt" beliebig oft gezeichnet oder wieder gelöscht werden, einfach dadurch, daß die Anweisungen "fenstmitt.Zeigen" oder "fenstmitt.Loeschen" zur Ausführung gebracht werden. Es braucht also kein weiterer Bezug auf die Koordinaten genommen zu werden. Der Rahmen existiert unter dem Namen

"fenstmitt". Ob er sichtbar oder unsichtbar ist, sind nur verschiedene Erscheinungsformen des Objekts "fenstmitt". Dies als kurze Einführung in die OOP. Eine weiter- und tiefergehende Darstellung des Sachverhalts befindet sich in (2*).

3.3.4. Das Programm example.pas

Alle Messungen am WÖTRA B außer der Durchgangsprüfung werden mit einem IEC-Bus gesteuerten Multimeter vom Typ Fluke 8842A durchgeführt. Um dieses Multimeter komfortabel über den PC steuern zu können, steht eine Treibersoftware von Philips zur Verfügung, die alle benötigten Routinen in einem .LIB-File zur Verfügung stellt. Hier zeichnete sich aber ein großes Problem ab. Dieses Library-File war auf die Verwendung mit dem MS-Pascal-Compiler zugeschnitten. MS-Pascal stellt einen sogenannten Mehr-Pass-Compiler dar, der im ersten und zweiten Durchlauf zunächst ein linkbares .OBJ-File erzeugt. Erst durch das Linken mit den benötigten Library-Files ensteht ein ausführbares .EXE-Programm. Turbo-Pascal hingegen ist ein Ein-Pass-Compiler, d.h. er erzeugt in einem Durchlauf ein ausführbares Programm. Das hat zwar den Vorteil, daß der Vorgang der Compilierung sehr schnell geht, dafür muß man mit dem Nachteil leben, daß keine Library-Files mit eingebunden werden können. Turbo-Pascal läßt eine Modularisierung von Programmen nur über das Unit-Konzept zu. Die einzige Möglichkeit zur Einbindung von bereits compilierten Programmen stellt die Compiler-Direktive "$I" dar, mit der .OBJ-Files, die im Intel-Format vorliegen, verarbeitet werden. Über diesen Umweg wurde auch versucht, das Problem zu lösen.
Hierzu wurde mit Hilfe des Microsoft Macro Assemblers (MASM 5.0) das .LIB-File der Treibersoftware in die einzelnen .OBJ-Files aufgeschlüsselt. Für diesen Zweck stellt der MASM die Bibliotheksverwaltung "LIB" zur Verfügung. Nachdem dies geschehen war, versuchte ich, die erhaltenen .OBJ-Da-

(2*) siehe Literaturverzeichnis
Handbücher zu Turbo-Pascal 5.5

teien mit der Anweisung "$I" in meine Programme einzubinden. Dies scheiterte jedoch daran, daß Turbo-Pascal ein anderes Datenformat für die Objekt-Files benötigt, als es der MASM erzeugt. Deshalb wurde diese Lösungsmöglichkeit wieder verworfen. Ich ging deshalb zu folgender Vorgehensweise über : das eigentliche Programm zur Steuerung des Multimeters und zur Durchführung der Messung ist in MS-Pascal geschrieben und wird zu einem ausführbaren Programm compiliert/gelinkt. Soll nun eine Messung durchgeführt werden, so wird vom Turbo-Pascal-Programm aus dieses Meßprogramm mit Hilfe der Exec-Prozedur aufgerufen. Die Schnittstelle zwischen den beiden Programmen stellen zwei Dateien dar, eine zur Übertragung der gewünschten Meßart an das MS-Pascal-Programm, und eine, um den resultierenden Meßwert an das aufrufende Turbo-Pascal-Programm zurückzuliefern.
Diese Handhabung der Messung hat den Nachteil, daß sie relativ langsam ist, da für jeden Meßwert vier Festplattenzugriffe erfolgen müssen und das Laden des Meßprogramms beim Aufruf durch die Exec-Prozedur viel Zeit in Anspruch nimmt. Da jedoch der Großteil der mit dem WÖTRA ausgeführten Messungen aus Durchgangsprüfungen bestehen, welche durch den A/D-Wandler-Kanal realisiert werden, kann der erhöhte Zeitbedarf bezogen auf die Gesamtdauer einer Baugruppenprüfung vernachlässigt werden.

3.3.4.1. Prozedur Messartlesen

Diese Prozedur dient dazu, die Art der Messung aus der Datei "paramet.dat", die vom aufrufenden Turbo-Pascal-Programm aus beschrieben wird, zu lesen. Um dies zu erreichen, wird zuerst der Variablen "f", die als "File of String" definiert wurde, der Pfad und Dateiname von "paramet.dat" auf der Festplatte des Rechners zugewiesen. Dann wird sie mit "Reset" zum Lesen geöffnet und die Meßart ermittelt. Die Close-Anweisung beendet dann die Prozedur vorschriftsmäßig.

3.3.4.2. Prozedur Messwertschreiben

Ist der Meßwert vom Multimeter ermittelt worden, so muß er zur korrekten Übergabe an das aufrufende Programm in der Datei "messwert.dat" abgespeichert werden. Hierzu wird wieder der Pfad und Dateiname von "messwert.dat" der Variablen "f" zugeordnet, die diesmal als "File of Real" definiert wurde. Ist dies erfolgt, so wird die Datei zum Schreiben geöffnet und der Meßwert abgespeichert. Mit "Close" wird die Datei geschlossen.

3.3.4.3. Das Hauptprogramm

Es wertet die benötigten Dateien aus und führt die eigentliche Messung durch.
Die Prozedur "Drivers_Initialisation" initialisiert, wie der Name schon sagt, die IEC-Bus-Karte im Rechner und das Meßgerät. Anschließend wird ermittelt, welche Art von Messung durchgeführt werden soll und entsprechend in einer IF-THEN-ELSE Abfragekette das Multimeter eingestellt. Mit "Set_Range" wird die Bereichswahl des Meßgeräts auf Automatik eingestellt. Die Prozedur "Set_Settling_Time(DMM1,Yes)" bestimmt, daß das Multimeter nach einer Aufforderung zur Durchführung einer Messung noch eine bestimmte Zeit wartet, bis auch alle Einstellvorgänge mit Sicherheit abgeschlossen sind. Wird dies nicht programmiert, so kann es zu erheblichen Ungenauigkeiten bei der Messung kommen, wie festgestellt wurde. Es folgt mit "Set_Trig_Source" noch eine Prozedur, die die Triggerquelle des Multimeters auf ein externes Signal von der IEC-Bus-Karte setzt. Mit dem Befehl "measure(DMM1,messwert)" wird die eigentliche Messung durchgeführt und anschließend in die Meßwertdatei geschrieben.
Sämtliche Prozeduren und Konstanten zur Einstellung des Digitalmultimeters sind in der Datei "Drivers.Inc", die zu Beginn des Programms eingebunden wurde, vordefiniert. Diese Datei ist Bestandteil der Treibersoftware zur IEC-Bus-Karte.

3.3.5. Das Include-File fenster.pas

Dieses Include-File wurde aus (3*) übernommen. Es dient dazu, beliebige Fenster am Bildschirm aufzumachen und den Hintergrund abzuspeichern, so daß beim Schließen des Fensters der alte Bildschirminhalt wieder restauriert wird. Um dies zu erreichen, wird vom Programm direkt auf den Bildschirmspeicher des Rechners zugegriffen.
Das Öffnen eines Fensters erfolgt mit der Prozedur "OpenWindow" unter Angabe der Eckpunkte des benötigten Fensters, das Schließen mit "CloseWindow". Da eine genauere Erläuterung des Programms hier zu weit führen würde, soll an dieser Stelle nur auf den entsprechenden Artikel verwiesen werden.

3.3.6. Das Unit helpme.pas

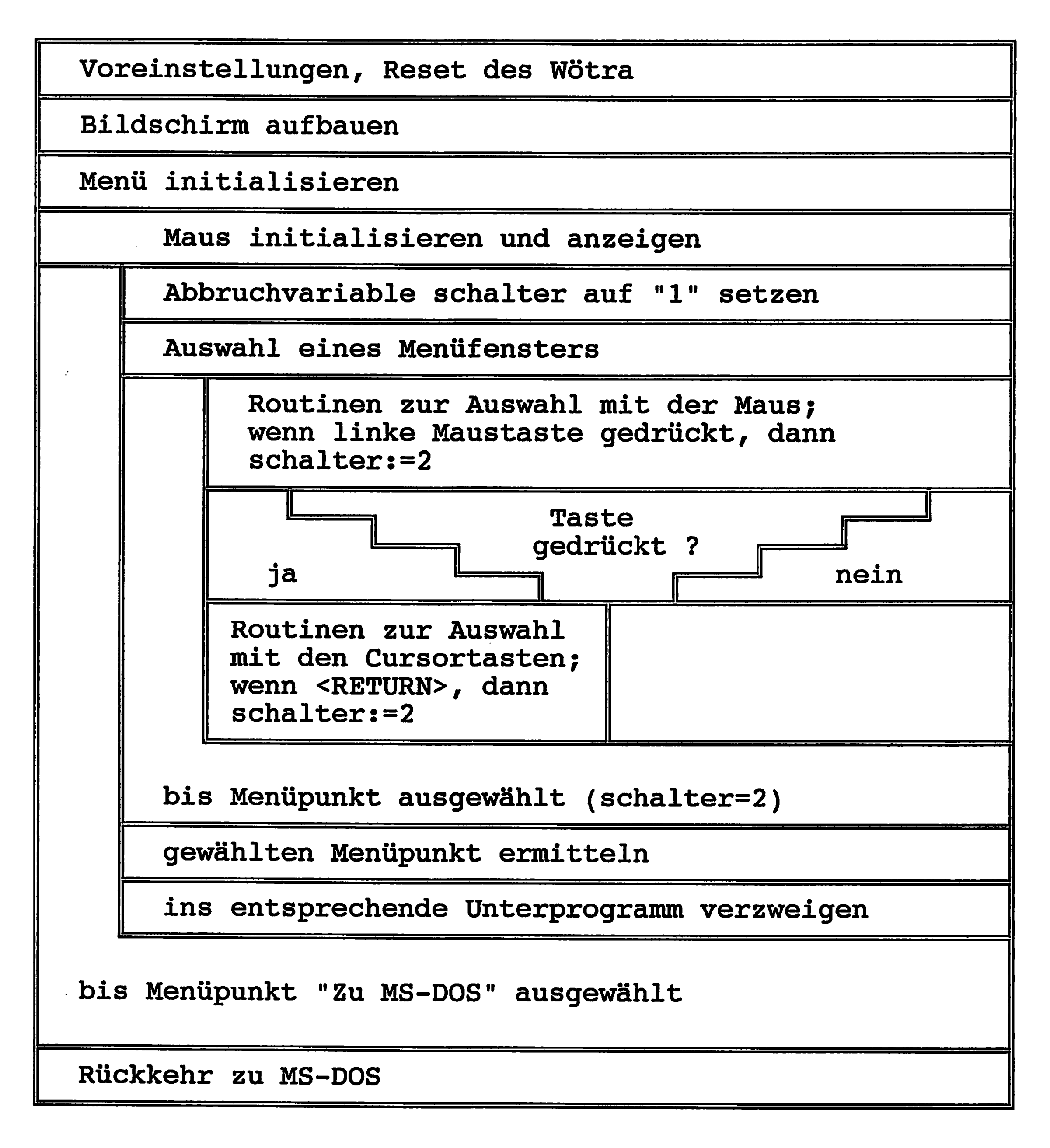
In diesem Unit sind alle Prozeduren zusammengefaßt, die dem Benutzer über <F1> erreichbare Hilfe-Funktionen zur Verfügung stellen.
Ihr Aufbau ist stets der Gleiche. Es wird ein Bildschirmfenster mit schwarzer Schrift auf grauem Grund eröffnet, welches in der oberen Umrandung den Namen des Menüpunkts enthält, zu dem Hilfe angefordert wurde. Dann wird durch eine Reihe von WriteLn-Anweisungen der eigentliche Hilfstext ausgegeben. Anschließend wird solange in einer Einleseschleife verweilt, bis die <ESC>-Taste gedrückt wurde. Bevor dann die Prozeduren verlassen werden, wird noch das Hilfe-Fenster geschlossen und der ursprüngliche Bildschirminhalt wieder hergestellt.
Auf die Prozeduren im Einzelnen soll hier nicht eingegangen werden, da sie sich aufgrund der Fülle der in ihnen enthaltenen WriteLn-Anweisungen sowieso von selbst erklären.

(3*) siehe Literaturverzeichnis unter "Turbo-Fenster auf dem PC"

3.4. Die Benutzeroberfläche

Die Benutzeroberfläche ist der zentrale Kern der gesamten WÖTRA-Software. Von ihr aus werden die Prüfdateien geladen, die Prüfung gestartet sowie alle Test- und Meßprogramme aufgerufen. Eine Hauptforderung an das Programm war die möglichst einfache Bedienung, d.h. sowohl durch die Maus als auch durch die Cursortasten. Die Anzeige der erreichbaren Funktionen sollte über Pull-Down-Menüs erfolgen. Hat der am Gerät arbeitende Prüfer einmal Schwierigkeiten mit der Bedienung, so sollte er durch Drücken der F1-Taste sofort eine kurze Online-Hilfe erhalten.
Wie bereits gesagt, ist das Menüprogramm für die parallele Bedienung durch die Maus und die Tastatur vorgesehen. Um das zu erreichen, wurde folgende Vorgehensweise gewählt: sämtliche Menüpunkte, die zur Auswahl stehen sollen, werden in einem Feld abgespeichert, in dem ihre Indizes entsprechend ihrer späteren Position am Bildschirm gewählt werden. An der Stelle, an der der Textcursor steht, wird der Menüpunkt durch inverse Farbgebung hervorgehoben. Dadurch wird erreicht, daß immer erkennbar ist, wo man sich im Menü gerade befindet. Wird nun Cursor-Auf oder Cursor-Ab betätigt, so wird, je nach Richtung der Taste, einfach der Menüpunkt an der alten Stelle des Textcursors wieder in den ursprünglichen Farben dargestellt, und an der Stelle des neuen wieder entsprechend invertiert. Beim Drücken von Cursor-Links oder Cursor-Rechts wird das momentan aktive Pulldown-Menü geschlossen, und, je nach gewünschter Richtung, das links oder rechts davon liegende Fenster geöffnet.
Um jetzt noch eine parallele Bedienbarkeit durch die Maus zu erhalten, muß bei jeder Betätigung einer Cursor-Steuertaste auch der Mauscursor an die neue Stelle nachgeführt werden. Umgekehrt muß auch bei einer Positionsänderung im Menü mit Hilfe der Maus der Textcursor auf die neue Stelle plaziert werden. Wie dies im Einzelnen realisiert wurde, wird bei der Beschreibung der Prozeduren noch genauer zur Sprache kommen.

3.4.1. Das Hauptprogramm



3.4.1.1. Struktogramm zum Hauptprogramm

3.4.1.2. Programmbeschreibung

Das Hauptprogramm der Benutzeroberfläche beginnt mit den Prozeduren "Port_Programmieren" und "Relais_Reset". Sie schalten den 8255-Baustein der Portkarte auf "Ausgabe" und versetzen den WÖTRA in den Grundzustand, d.h. alle Relais fallen ab. "Eingangsmenue" gestaltet die Bildschirmmaske, "Cur_Off" schaltet den blinkenden Cursor aus. Die Prozedur

"Begruesung" bringt die Eingangsmeldung des Programms auf den Bildschirm. In "Menu_Init" werden die einzelnen Menüpunkte in einem Feld ("dateiarray") abgespeichert, welches dann von den weiteren Prozeduren zur Bildschirmgestaltung verwendet wird. Die drei nachfolgenden Befehle setzen die Maus in den Grundzustand, verlangsamen ihre Bewegung und machen sie am Bildschirm sichtbar. Die darauf folgende Programmschleife wird solange ausgeführt, bis der Benutzer einen kontrollierten Programmausstieg über den Menüpunkt "Zu MS-DOS" wählt. Innerhalb dieser Schleife wird zuerst wieder der Benutzerbildschirm aufgebaut und das Abbruchkriterium für die nachfolgende Auswahlschleife in den Grundzustand versetzt ("schalter=1"). Anschließend wird die Auswahlschleife abgearbeitet. Sie wird solange durchlaufen, bis entweder für einen Menüpunkt die linke Maustaste oder <RETURN> gedrückt wurde (ist dies der Fall, so wird die Variable "schalter" auf 2 gesetzt und damit die REPEAT-Schleife verlassen). Innerhalb der Auswahlschleife wird die Prozedur zur Auswahl eines Menüpunktes mit der Maus solange in einer Endlosschleife wiederholt, bis durch Drücken einer Taste in den Auswahlblock für die Cursortasten verzweigt wird.
Wurde ein Menüpunkt durch Drücken von <RETURN> oder der linken Maustaste ausgewählt, so läuft das Programm in eine IF-THEN-ELSE-Kaskade, in der der gewünschte Menüpunkt ermittelt wird. Abhängig von diesem wird die Variable "wahl" auf einen bestimmten Wert gesetzt und in der sich anschließenden Case-Anweisung in das entsprechende Unterprogramm verzweigt.

3.4.2. Prozedur Menu_Init

Durch diese Prozedur wird das Feld "dateiarray" initialisiert, in dem alle Punkte des Menüs als Strings abgespeichert sind. Die Feldindizierung ist so ausgelegt, daß die erste Zahl die Nummer des Fensters bezeichnet, in dem der Punkt aufgeführt wird ("Prüfdatei" entspricht 1, "Aktionen" entspricht 2, usw.) und die zweite Zahl die Position des

Menüpunkts innerhalb des Fensters von oben (Nummerierung in Anlehnung an die y-Koordinaten des Textcursors).
Weiterhin wird die maximale Anzahl von Menüpunkten für jedes Menüfenster im Feld "dateiarraymax" abgelegt. Dieses Feld ist dann wichtig, wenn z.B. bei der Menüpunktauswahl die Taste <Cursor auf> gedrückt wird, obwohl schon das obere Ende des Menüfensters erreicht ist. Durch den in "dateiarraymax" abgelegten Wert kann jetzt ganz einfach zum untersten Menüpunkt des Fensters verzweigt werden. Entsprechendes passiert, wenn <Cursor-Ab> gedrückt wird, obwohl schon das untere Ende des Menüfensters erreicht ist. Hier wird dann nach oben auf den ersten Menüpunkt gesprungen. Dieses Verfahren der Menüsteuerung ohne feste Endemarke innerhalb eines Fensters wird als "wrap-around-Technik" bezeichnet.

3.4.3. Prozedur Cur_Off

Sie ist dafür zuständig, den Cursor unsichtbar zu machen. Im Gegensatz zu "Cursor_Off" aus dem Unit "global.pas", die ja dasselbe erledigt, verändert "Cur_Off" nicht die aktuelle Position des Cursors. Dies ist deshalb von entscheidender Bedeutung, da das gesamte Menüprogramm Bezug auf die Bildschirmkoordinaten des Cursors nimmt. Würde nun beim Abschalten seine Position verändert werden, so käme die ganze Bildschirmsteuerung durcheinander.
Einen Ausweg aus diesem Dilemma bietet der BIOS-Interrupt 10h, der für die Grafikkarten des PCs zuständig ist. Die Funktionsnummer 1 diese Interrups ist dafür vorgesehen, die Erscheinungsweise des Cursors zu definieren. Dabei muß im CH-Register die Anfangs- und im CL-Register die Endzeile stehen, in der er sichtbar sein soll. Wählt man nun die Anfangszeile größer als die Endzeile, so verschwindet er vom Bildschirm. Genau dies ist in der vorliegenden Routine gemacht worden.

3.4.4. Prozedur Cur_On

Funktionsweise wie "Cur_Off", macht den Cursor aber wieder sichtbar. Hier muß die Anfangszeile kleiner als die Endzeile gewählt werden. Für den normalerweise von MS-DOS benutzten einzeiligen Cursor am unteren Rand der Zeile betragen diese Werte 11 und 12.

3.4.5. Prozedur Oeffne_Fenster

Diese Prozedur ist dafür verantwortlich, daß das vom Benutzer gewünschte Pull-Down-Menü geöffnet wird. Die Variable "k", die übergeben wird, gibt dabei die Nummer des Fensters an. Da insgesamt vier Pull-Down-Menüs vorhanden sind, muß k im Bereich von 1 bis 4 liegen. Der prinzipielle Ablauf ist für jedes k gleich, so daß hier stellvertretend nur der für k=1, also dem Menüfenster "Prüfdatei", beschrieben wird.
Zuerst werden Texthinter- und Vordergrund auf Grau und Schwarz gesetzt, damit sich die Menüfenster auch optisch entsprechend vor dem Rest des Bildschirms abheben. Dann wird mit der Prozedur "OpenWindow" aus dem Include-File "fenster.pas" ein Textfenster am Bildschirm aufgemacht. Anschließend muß der Texthinter- und Vordergrund erneut gesetzt werden, da sich herausstellte, daß die Prozedur "OpenWindow" die Einstellungen der Farben ändert. Ist dies geschehen, so werden die einzelnen Menüpunkte in das Textfenster geschrieben, wobei jedoch der erste in Weiß auf schwarzem Grund dargestellt wird, um dem Bediener auch optisch zu signalisieren, auf welchem Menüpunkt er sich gerade befindet.
Ist die Befehlsfolge für das entsprechende k abgearbeitet, wird noch die zu Beginn unsichtbar gemachte Maus (dies ist notwendig, da andernfalls an der Stelle, an der sich der Mauscursor befand, beim Schließen des Fensters farbige "Löcher" zurückbleiben) wieder am Bildschirm gezeigt und die Prozedur verlassen.

3.4.6. Prozedur Menue_Hilfe

Falls der Bediener des Programms einmal vergessen hat, was die einzelnen Menüpunkte zu bedeuten haben, kann er jederzeit mit <F1> eine Online-Hilfe aufrufen.
Im Programm wird dies mit der Prozedur "Menu_Hilfe" verwirklicht. Durch eine IF-THEN-ELSE-Abfragekette wird ermittelt, in welchem Menüpunkt sich der Maus- bzw. Textcursor gerade befindet, d.h. welcher Punkt aktiviert ist. Dann wird entsprechend dieser Position eine von 12 Hilfsfunktionen aufgerufen. Die Hilfsfunktionen befinden sich alle im Unit "Helpme.pas".

3.4.7. Prozedur Maus_Auswahl

In dieser sehr umfangreichen Prozedur wird die Maussteuerung bei der Auswahl eines Menüpunktes realisiert.
Zu Beginn wird wieder der Mauszeiger unsichtbar gemacht, um störende Farbüberschneidungen beim Auf- und Zumachen der Fenster zu vermeiden. Anschließend folgt eine Schleife, die solange durchlaufen wird, bis entweder eine Taste oder der linke Mausknopf zur Auswahl eines Punkts gedrückt wird. Innerhalb dieser Schleife sind vier große Abfrageblöcke eingebettet, für jedes der vier Menüfenster ("Prüfdatei", "Aktionen", "Messen", "Test") einer. Ist nun ermittelt, in welchem dieser Fenster der Cursor sich befindet, so wird durch eine ganze Reihe von IF-Abfragen die genaue Position innerhalb des Fensters herausgefunden. Hat man das erreicht, so wird der normale Textcursor entsprechend nachgeführt.
Um nun dem Bediener den Wechsel eines Menüpunktes innerhalb des gerade aktivierten Menüfensters zu verdeutlichen, wird in den Variablen "switch" und "switch1" jeweils der alte und der neue Standort innerhalb des Fensters abgespeichert, so daß der alte Menüpunkt in den ursprünglichen und der neue in den invertierten Farbwerten dargestellt werden kann. Dies geschieht jeweils am Ende der vier Abfrageblöcke.

In jeden der Blöcke ist noch eine Schleife integriert, die wieder solange durchlaufen wird, bis eine Taste oder der linke Mausknopf gedrückt wird. Als zusätzliches Abbruchkriterium gilt noch das Wechseln des Menüfensters, was durch die Abfrage der x-Koordinate der Maus erkannt wird.
Die beschriebenen Vorgänge laufen wie bereits gesagt solange ab, bis eine Auswahl getroffen wird, d.h. Drücken der linken Maustaste oder <RETURN>. In diesem Fall wird am Schluß der Prozedur in die Variable "auswahl" der gerade gewählte Menüpunkt übernommen und im bereits beschriebenen Hauptprogramm ausgewertet.
Erfolgte das Verlassen eines Auswahlblocks aufgrund der Tatsache, daß ein geöffnetes Fenster verlassen wurde, so wird der Mauscursor in den neuen Bildschirmbereich gesetzt, das alte Fenster geschlossen und das neue geöffnet. Einen Bezug zum aktuellen Menüfenster bildet die Variable "a", in der dessen Nummer abgespeichert ist.

3.4.8. Prozedur Cursor_Auswahl

Sie dient zur Menüsteuerung mit den Cursor-Tasten.
Diese Prozedur wird nur dann ausgeführt, wenn die Routine "Maus_Auswahl" durch Drücken einer der vier Cursor-Tasten verlassen wurde. Dies wird gleich zu Beginn überprüft. Stellt sich heraus, daß die gedrückte Taste die Return-Taste war, so wird der gerade aktive Menüpunkt übernommen, das Fenster geschlossen und zum Hauptprogramm zurückgekehrt.
War die gedrückte Taste nicht die Return-, sondern eine der vier Steuertasten für den Textcursor, so wird in einer Case-Anweisung auf den entsprechenden Auswahlblock verzweigt.
Im Anweisungsblock für <Cursor-Auf> wird zunächst abgefragt, ob das obere Menüende bereits erreicht ist. Wenn nicht, wird einfach der alte Menübalken gelöscht und der neue farblich hervorgehoben. Andernfalls wird auf den untersten Punkt des Menüfensters gesprungen und dieser markiert.

Wurde die Taste <Cursor-Ab> gedrückt, so muß die Überprüfung dahingehend erfolgen, ob das untere Ende des aktivierten Menüfensters erreicht ist. In diesem Fall wird dann zum obersten Punkt gesprungen. Durch diese Auswertung der Cursor-Auf-Ab-Tasten wird der bereits angesprochene "wrap-around"-Effekt erzeugt.
Für den Fall, daß die Taste Cursor-Links oder -Rechts gedrückt war, wird die Variable "a", die ja immer die Nummer des gerade geöffneten Pull-Down-Menüs enthält, entweder erhöht oder erniedrigt. Um hier auch in horizontaler Richtung den "wrap-around"-Effekt zu erhalten, muß bei der In- oder Dekrementierung der Variablen "a" darauf geachtet werden, daß ihr Wert immer zwischen 1 und 4 liegt. Dies ist jeweils mit einer If-Abfrage gelöst worden. Anschließend wird dann das alte Menüfenster geschlossen und das neue geöffnet.
Nachdem dann noch der Mauszeiger auf die neue Position korrigiert worden ist, kann die Prozedur verlassen werden.

3.4.9. Prozedur Eingangsmenue

In diesem Programmteil werden Menü- und Statuszeile sowie das Arbeitsfenster auf dem Bildschirm erzeugt. Die Gestaltung wurde entsprechend der Turbo-Pascal Bedienoberfläche ausgeführt, d.h die Menü- und Statuszeile erscheinen grau, der Arbeitsbildschirm blau. Die Menü- und Statuspunkte wiederum werden in schwarz geschrieben, wobei der erste Buchstabe rot hervorgehoben wird. Dies wurde gemacht, um zu verdeutlichen, daß das entsprechende Menü anstatt mit der Maus oder der Funktionstaste <F10> auch durch Drücken des Anfangsbuchstabens erreicht werden kann.

3.4.10. Prozedur Begruesung

Diese Prozedur wird nur einmal zu Beginn des Programms aufgerufen. Da sich die WriteLn-Anweisungen sozusagen von selbst erklären, soll hier nicht genauer darauf eingegangen werden.

3.4.11. Prozedur Menue_Auswahl

In der durch die Prozedur "Eingangsmenue" aufgebauten Bildschirmmaske erfolgt beim Programmstart und bei der Rückkehr aus einem Unterprogrammpunkt die Auswahl eines Menüfensters. Diese Auswahlmöglichkeit wird von der Prozedur "Menue_Auswahl" zur Verfügung gestellt.
Der Bediener hat drei Möglichkeiten, ein Menüfenster zu erreichen: Anklicken des Menüpunktes mit der Maus, Drücken des rot hervorgehobenen Anfangsbuchstabens oder Drücken der Funktionstaste <F10>. Alle drei Möglichkeiten werden von "Menue_Auswahl" abgedeckt.
Hierzu wird zunächst wieder die Mausposition am Bildschirm abgefragt und der Variablen "zeichen" ein entsprechender Wert zugewiesen. Dieser Wert ist gleich dem ASCII-Code des Anfangsbuchstabens des gewünschten Menüpunktes. Die Abfrage der Mausposition erfolgt wieder solange, bis entweder der linke Mausknopf oder eine Taste gedrückt wird. Ist dies erfolgt, so wird in der folgenden Case-Anweisung abhängig vom Wert der Variablen "zeichen" ein Menüfenster am Bildschirm eröffnet, in welchem dann wie beschrieben die Auswahl eines Menüpunktes durch die Prozeduren "Maus_Auswahl" und "Cursor_Auswahl" erfolgen kann.
Einen Sonderfall bildet das Drücken der Funktionstasten <F1> oder <F10>. Mit <F1> wird am Bildschirm ein Hilfsfenster aufgemacht, in dem die wichtigsten Informationen über die Bedienung des Programms dem Benutzer zugäglich gemacht werden. Durch Drücken von <F10> wird das Menüfenster "Prüfdatei" geöffnet und direkt in die Menüpunktauswahl verzweigt.

3.5. Die Untermenüs

3.5.1. Das Menü Prüfdatei

3.5.1.1. Laden

Bildschirm vorbereiten (Menü-, Statuszeile usw.)	
Aufforderung, Dateiname direkt oder <F3> einzugeben	
Eingabe ?	
ASCII-Zeichen direkt	<F3>
eingegebenes Zeichen als Echo am Monitor	Directory-Funktion aufrufen (GetFileName *.dat)
Rest des Dateinamens einlesen	Datei auswählen
gewählte oder eingegebene Datei anzeigen	
mit der Wahl einverstanden ? ja	nein
schalter:=TRUE	schalter:=FALSE
wiederholen, bis schalter=TRUE	
in die Benutzeroberfläche zurückkehren	

Bevor eine Baugruppenprüfung durchgeführt werden kann, muß erst einmal die zugehörige Prüfdatei geladen werden. Da auch hier wieder die Hauptforderung lautete, den Vorgang für den Prüfer so einfach wie möglich zu gestalten, wurde eine Directory-Funktion ähnlich der von Turbo-Pascal realisiert. Das bedeutet, daß beim Aufruf des Menüpunkts "Laden" in der Benutzeroberfläche entweder die gewünschte Datei und ihre Pfadangabe direkt eingegeben werden kann oder durch Drücken der Funktionstaste <F3> eine Übersicht über alle vorhandenen Prüfdateien erscheint. In dieser Übersicht kann sich der Prüfer mit den Cursor-Steuertasten frei bewegen. Die jeweils angewählte Datei wird dabei durch einen grau

hinterlegten Balken markiert. Die Übernahme der Datei erfolgt dann durch Drücken von <RETURN>.
Aus zeitlichen Gründen wurde beschlossen, die sehr komfortable Directory-Funktion nicht selbst zu schreiben, sondern in der Literatur nach entsprechenden Routinen nachzuforschen. In (4*) wurde ich schließlich fündig. Die dort beschriebenen Units ("filename.pas", "director.pas", "windo.pas")gestatten es, durch einen einfachen Funktionsaufruf die Directory-Anzeige zu erzielen. Der Name der Funktion lautet "GetFileName". Mit Hilfe dieser Routine wurde dann der Ladevorgang für Prüfdateien programmiert.
Am Anfang des Units "Laden.pas" wird im Interface-Teil die Prozedur "Lade", die den gesamten Einlesevorgang für Prüfdateien enthält, für den Zugriff von anderen Programmen freigegeben.
Im Implementationsteil werden zuerst die benötigten Units eingebunden. Dies sind zum Einen die Standardunits von Turbo-Pascal und die Unit "global,pas", in der die Variable "pruefname", welche später den Pfad und Namen der ausgewählten Datei erhält, definiert ist. Die Variable wurde deshalb in "global.pas" definiert, um sie dann nach dem Ladevorgang im Unit "startpru.pas", welches die Prüfablaufroutinen enthält, unter gleichem Namen verwenden zu können. Dazu muß dann dort lediglich wieder das Unit "global" eingebunden werden. Zum Anderen ist in der USES Anweisung von "Laden.pas" noch das Unit "filename" enthalten, welches die Funktion für die Directory-Anzeige enthält.
Das File "fenster.pas" wurde wieder eingebunden, um schnelle Bildschirmfenster zur Verfügung zu haben.
Nun zur eigentlichen Prozedur "Lade":
zu Beginn wird wieder der Bildschirm in Menü- und Statuszeile sowie Arbeitsfenster aufgeteilt.
Dann folgt eine Schleife, die solange durchlaufen wird, bis der Bediener durch eine Bestätigung am Schluß des Programms seine Dateiwahl als richtig deklariert. Diese Schleife ist

(4*) siehe Literaturverzeichnis "Directorys unter Turbo-Pascal"

wichtig, um Fehleingaben möglichst einfach korrigieren zu können. In ihr wird dann ein Fenster geöffnet, und der Benutzer aufgefordert, entweder den Pfad und Dateinamen direkt einzugeben, oder mit der <F3>-Taste eine Übersicht anzufordern. Anschließend wird ein Zeichen von der Tastatur gelesen. Ist der Scancode der Taste Null, so wurde die Funktionstaste gedrückt und der Anweisungsblock für das komfortable Directory-Programm kommt zur Ausführung. Hierzu wird mit "GetDir" zuerst das aktuelle DOS-Verzeichnis gespeichert und anschließend mit "GetFileName" die Dateiübersicht am Bildschirm erzeugt. Wurde dann mit <RETURN> eine Datei ausgewählt und der Variablen "pruefname" zugeordnet, so wird wird mit "ChDir" wieder ins ursprüngliche Verzeichnis zurückgekehrt.
Wurde die <F3>-Taste nicht gedrückt, so befindet sich der Bediener im Programmteil zur manuellen Dateieingabe. Der zuerst eingegebene Buchstabe und der Rest der Eingabe werden zur Variablen "pruefname" zusammengesetzt.
Nun wird unabhängig davon, ob die Prüfdateiauswahl per Hand oder über die Verzeichnisanzeige erfolgte, die ausgesuchte Datei und ihre Pfadangabe noch einmal am Bildschirm angezeigt. Dies hat den Sinn, daß der Prüfer noch einmal Gelegenheit hat, seine Wahl zu kontrollieren. Ist er zufrieden, so gelangt er durch Drücken von <j> wieder in die Benutzeroberfläche zurück. Drückt er hingegen <n>, so wird wieder an den Programmanfang verzweigt, und der ganze Programmablauf erfolgt von Neuem.

3.5.1.2. Erstellen

Dieser Programmteil wurde nicht von mir, sondern im Rahmen einer dritten Diplomarbeit über das Baugruppenprüfgerät WÖTRA B von Markus Kölbl, Fachhochschule Nürnberg, bearbeitet. Sein Programm wurde nicht in Form eines Units in die Benutzeroberfläche eingebunden, sondern wird über die Exec-Prozedur, die Turbo-Pascal zur Verfügung stellt, vom Menüprogramm aus aufgerufen. Damit dies fehlerfrei erfolgen kann, muß sich das Programm diwö.exe im Verzeichnis "c:\tp5\" befinden.
Eine detailierte Beschreibung des Programms "diwö.exe", sowie eine Bedienungsanleitung dazu findet man in (5*). Deshalb soll an dieser Stelle nicht näher darauf eingegeangen werden.

3.5.1.3. Schreibe nach
3.5.1.4. Drucken

Die Routinen für die Menüpunkte "Schreibe nach" und "Drucken" waren nicht Bestandteil meiner Diplomarbeit. Sind sie einmal realisiert, so können sie im Hauptprogramm der Benutzeroberfläche "pruefpr4.pas" in der Case-Anweisung unter Punkt drei und vier eingefügt werden. Die entsprechenden Stellen sind im Listing gekennzeichnet.

(5*) siehe Literaturverzeichnis Diplomarbeit Markus Kölbl

3.5.2. Das Menü Aktionen

3.5.2.1. Start

Dies ist das Hauptprogramm zur Baugruppenprüfung am WÖTRA B. Wird dieser Menüpunkt ausgewählt, so verzweigt das Programm zur Prozedur "Pruefung" des Units "startpru.pas", in dem alle Prozeduren zur Baugruppenprüfung enthalten sind. Das WÖTRA-Konzept wurde ja von Anfang an darauf ausgelegt, die Prüfdateien für die verschiedenen Baugruppen mit einem Dateigenerator zu erstellen und durch ein für alle Baugruppen gleichbleibendes Prüfprogramm diese Dateien auswerten zu lassen. Dieses Auswerteprogramm soll nun beschrieben werden.

3.5.2.1.1. Der Aufbau der Prüfdateien

Zu Beginn der Softwareerstellung wurde mit dem Diplomanden Markus Kölbl, Fachhochschule Nürnberg, der für die Entwicklung des Prüfdateigenerators zuständig war, festgelegt, wie die Prüfdateien aufgebaut sein sollten. Das Ergebnis dieser Besprechung war, daß jede Prüfdatei als File of Record organisiert wird. Jedem Prüfschritt, der zur Ausführung kommen soll, wird ein solches Record zugeordnet. Es wird im Unit "global.pas" definiert, und hat folgenden Aufbau:
Im Feld "koppelpunkt", das acht Plätze aufweist, werden die Nummern der Koppelpunkte abgelegt, welche mit Prüfspannungen beschaltet werden sollen. Dabei ist für jede zu schaltende Spannung ein Koppelpunktpaar erforderlich. Dies hat zur Folge, daß am WÖTRA B maximal vier Spannungen an die Ausgänge geschaltet werden können. Dies mag als erhebliche Einschränkung der Prüfmöglichkeiten angesehen werden. Da der WÖTRA B jedoch nur als Unterstützung für die bereits vorhandenen Baugruppenprüfgeräte der ersten Generation dienen soll und bei diesen nie mehr als vier Prüfspannungen gleichzeitig vorkommen, spielt diese Einschränkung in der Praxis keine Rolle. Es gibt noch zwei Sonderfälle im Feld

"koppelpunkt": steht im ersten Paar die Null, so bedeutet dies, daß der Leistungspunkt geschaltet werden soll. Steht in einem Koppelpunktpaar die Zahl 999, so sollen keine weiteren Spannungen angelegt werden.
In den nächsten drei Datenfeldern des Records "einzelschritt" stehen drei Werte vom Typ REAL, in denen die Vorgaben zur Durchführung von Messungen am WÖTRA B abgelegt sind. Dies sind im Einzelnen der Sollwert, der Wert für die zulässige Toleranz der Messung nach Oben sowie die Toleranz nach Unten.
Im Feld "spannung" von "einzelschritt" ist gespeichert, welche Spannungen an die verschiedenen Koppelpunktpaare anzulegen sind. Dabei enthält "spannung[1]" den Wert für das erste Koppelpunktpaar, "spannung[2]" den Wert für das zweite usw.. Es sind folgende Möglichkeiten für den String "spannung" vorgesehen: 220V Wechselspannung, 110V Wechselspannung, 24V Wechselspannung, 24V Gleichspannung, 0-100V Gleichspannung sowie der Leistungspunkt.
Die nächsten drei Felder des Records enthalten die Variable "pruefschritt", in der die Nummer des gerade zu bearbeitenden Prüfdurchgangs gespeichert ist und die Werte für die beiden Knotennummern, an denen eine Messung erfolgen soll. Sie liegen wieder im Bereich von 1-400. Für die Meßknoten wurde jedoch noch eine zusätzliche Vereinbarung getroffen. Enthalten sie den Wert 999, so soll keine Prüfung durchgeführt, sondern ein Haltepunkt im Prüfprogramm gesetzt werden. Dies ist deshalb erforderlich, da das Prüfprogramm selbstständig Schritt für Schritt abgearbeitet wird, solange kein Fehler auftaucht. Es kann jedoch die Situation auftreten, daß das Programm anhalten soll, ohne daß ein Fehler in der Baugruppe ist. Solch ein Ereignis ist z.B. der Programmstart, in dem dem Bediener kurze Informationen über die gerade zu prüfende Baugruppe mitgeteilt werden sollen. Noch wichtiger ist der Zeitpunkt vor dem Aufschalten von 220 V auf die Baugruppe. Hier muß auf jeden Fall das Programm anhalten, um den Prüfer zu warnen, daß von nun an eine lebensgefährliche Spannung am Prüfling anliegt. Auch bei einer Strommessung muß dem

Bediener durch einen Programmstop der Hinweis gegeben werden, daß die Anschlußkabel für das Meßgerät umzustecken sind. All diese Fälle werden durch das Setzen eines Haltepunkts mit der Meßknotennummer 999 verwirklicht. Der für den jeweils zutreffenden Fall erforderliche Hinweis an den Bediener muß dann in die nachfolgend beschriebene Kommentar- oder Warnungszeile eingegeben werden.
Diese beiden Zeilen werden im Record als String definiert und können jeweils maximal 128 Zeichen für Meldungen an den Prüfer beinhalten. Solche Meldungen können wie Oben beschrieben Warnungen, oder auch Kommentare und Hinweise zur Fehlersuche an der Baugruppe sein. Sie werden bei jedem Prüfschritt mit ausgegeben und sind auch bei einem Programmstop aufgrund eines Fehlers in der Baugruppe am Monitor sichtbar.
Zum Schluß des Records ist noch die Variable "messart" vorhanden. In ihr steht, wie der Name schon besagt, die Art der Messung, die bei einem Prüfschritt ausgeführt werden soll. Hierbei gibt es folgende Möglichkeiten:
Nichdurchgangsprüfung, Durchgangsprüfung, Gleichspannung und -strom, Wechselspannung und -strom sowie 2-Draht Widerstandsmessung.
Da für die meisten Prüfschritte das Record nicht voll beschrieben werden muß, stehen für den Fall einer Nichtbenutzung folgende Werte in den einzelnen Feldern:

"koppelpunkte" : "999"
"sollwert", "toleranzoben", "toleranzunten" : "0"
"spannung" : "-- --"
"messknoteneins", "messknotenzwei" : "0"
"kommentar", "warnung", "messart": Leerzeichen
"messart": Leerzeichen

3.5.2.1.2. Prozedur Messartschreiben

Diese Prozedur stellt das Gegenstück zu "Messartlesen" aus dem MS-Pascal-Programm "example.pas" dar. Mit ihrer Hilfe wird die Datei "paramet.dat" mit der Art der Messung, die

durchgefürt werden soll, beschrieben. Hierzu wird zuerst der Variablen "f" der Dateiname und Pfad zugeordnet, anschließend die Datei zum Beschreiben geöffnet, dann die Meßart in die Datei geschrieben und mit der Prozedur "Close" wieder ordnungsgemäß geschlossen.

3.5.2.1.3. Prozedur Messwertlesen

Sie ist das Gegenstück zu "Messwertschreiben" aus "example.pas". Ihre Aufgabe ist die Übernahme des durch das Digitalmultimeter gemessenen Wertes aus der Datei "messwert.dat". Dazu wird der Variablen "f" wieder der Dateiname und Pfad zugeordnet, die Datei zum Lesen geöffnet, der Meßwert eingelesen und anschließend die Datei wieder geschlossen.

3.5.2.1.4. Prozedur Knoten_Reset

Die Prüfung einer Baugruppe mit dem WÖTRA B erfolgt Schritt für Schritt, wobei in jedem Prüfabschnitt verschiedene Spannungen an den Prüfling geschaltet werden. Damit es dabei nicht zu Kurzschlüssen kommen kann, muß nach jedem Prüfschritt ein kleiner Reset des WÖTRA erfolgen, bei dem alle Knoten, welche mit Spannungen beaufschlagt waren, wieder spannungsfrei werden. Dies erledigt die Prozedur "Knoten_Reset". Dazu wird eine Schleife von eins bis acht durchlaufen und jedesmal an den entsprechenden Koppelpunkt eine Null ausgegeben, was einem Rücksetzen aller Relais des Knotens entspricht. Anschließend werden noch die beiden Meßknoten zurückgesetzt und ein Reset der Umschaltplatinen durchgeführt. Die Prozedur hierfür, "Umschaltung_Reset", befindet sich im Unit "global.pas" und ist im Kapitel 3.3.2.6 genauer erläutert.

3.5.2.1.5. Prozedur Testspannung

Sie hat die Aufgabe, den A/D-Wandler-Wert, der mit der Prozedur "Spannung_Lesen" (siehe 3.3.2.4) ermittelt wurde, in

den tatsächlichen Spannungswert umzurechnen. Genaueres über die dazu notwendige Skalierung befindet sich im Handbuch zu der universellen Portkarte der Firma Impec.

3.5.2.1.6. Prozedur Eingangs_Maske

In dieser Prozedur wird die Bildschirmmaske für den Prüfablauf erstellt. Dazu erfolgt zunächst die nun bereits oft beschriebene Einteilung in Menü- und Statuszeile sowie Arbeitsfenster.
Dann wird die Überschrift und die Sachnummer der zu prüfenden Baugruppe ausgegeben. Die Sachnummer steht in der Variablen "ausgabename" und wird zu Beginn des Hauptprogramms aus dem Dateinamen ermittelt. Die Sachnummern der Baugruppen wurden deshalb als Namen für die Prüfdateien verwendet, weil damit ein eindeutiger Bezug zwischen Prüfling und Prüfdatei möglich ist.
Nach der Überschrift werden die Fensterrahmen für die Ausgabe der Kommentare und Warnungen während der Prüfung sowie für die Anzeige der Ist- und Sollwerte bei einer Messung auf dem Monitor gezeichnet. Dies hat den Sinn, daß der Prüfer bei einem Programmstop aufgrund eines Fehlers in der Baugruppe alle für eine Fehlersuche benötigten Werte am Bildschirm zur Verfügung hat.

3.5.2.1.7. Prozedur Warnung_Halt

Wie bereits angesprochen, gibt es verschiedene Situationen, in denen keine Messung, sondern nur die Ausgabe einer Warnung am Bildschirm erfolgen soll. Dazu wird in vier FOR-Schleifen die Warnung und der Kommentar aus dem aktuellen Prüfschrittrecord zeichenweise ausgegeben. Hierauf wird ein Bildschirmfenster mit rotem Hintergrund zur Untermauerung des Warneffekts eröffnet, in dem der Bediener aufgefordert wird, die Warnung und den Kommentar zu beachten und nach erfolgter Registrierung derselben durch Drücken der <RETURN>-Taste den Prüfvorgang fortzusetzen. Hat er dies

getan, so wird das Warnungsfenster wieder vom Bildschirm gelöscht und mit dem nächsten Prüfschritt begonnen.

3.5.2.1.8. Die Prozeduren Fehler_Routine1-3

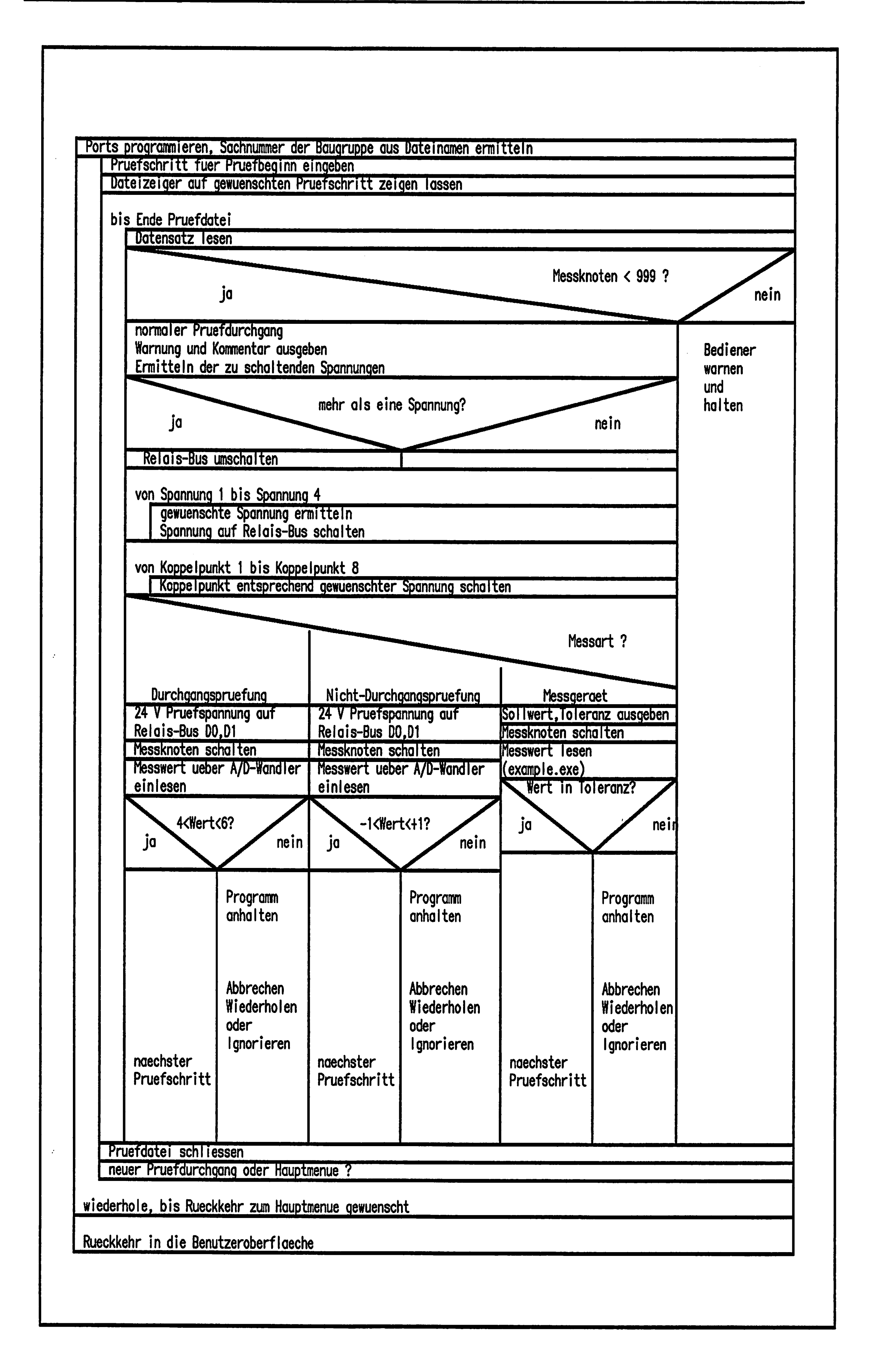
Tritt während des Prüfablaufs ein Fehler auf, so muß das Programm anhalten und dem Prüfer die Fehlerart anzeigen. Hierfür sind die drei vorliegenden Prozeduren vorgesehen. Sie sind alle gleich aufgebaut, lediglich in der Fehlermeldung unterscheiden sie sich.
Zu Beginn wird ein Fenster am Bildschirm aufgebaut, welches wieder rot hinterlegt wird. Darin wird dem Prüfer mitgeteilt, um welche Art Fehler an welchen Koppelpunkten es sich handelt. Anschließend wird er vor die Wahl gestellt, den Prüfvorgang abzubrechen, den Prüfschritt zu wiederholen oder den Fehler zu ignorieren (dies ist vor allem dann sinnvoll, wenn z.B. ein Meßwert nur geringfügig außerhalb des Toleranzbereichs liegt). In einer nachfolgenden Case-Anweisung wird dann die Eingabe des Prüfers ausgewertet.
Im später beschriebenen Prüfhauptprogramm laufen die Prozeduren "Fehler_Routine" in einer REPEAT-Schleife ab, an deren Ende die Variable "fehlercode" abgefragt wird. Wählt der Bediener innerhalb einer Fehlerroutine die Option Programmabbruch, so wird diese Variable auf den Wert "2" gesetzt und das Programm mit "Exit" verlassen. Entscheidet er sich für die Ignorierung des Fehlers, so wird "fehlercode" auf "0" gesetzt, was zur Folge hat, daß das Programm mit der Prüfung fortfährt, als sei kein Fehler aufgetreten. Drückt er <W> für Wiederholen, so erhält "fehlercode" den Wert "1". Dies bewirkt, daß die übergeordnete Programmschleife und damit der aktuelle Prüfschritt erneut zur Ausführung kommt.
Wenn die Fehlerroutine abgearbeitet ist, wird das Fehlerfenster vom Bildschirm entfernt und die alten Werte für die Farbgebung der Ausgaben am Bildschirm wieder hergestellt.

3.5.2.1.9. Prozedur Messung_Durchfuehren

Diese Prozedur wird benötigt, wenn in einem Prüfschritt eine Messung mit dem Digitalmultimeter durchgeführt werden soll.
Dazu wird mit der Prozedur "Messartschreiben" die gewünschte Meßart in die Datei "paramet.dat" geschrieben und auch im Protokollfenster am Monitor ausgegeben. Anschließend wird mit der Exec-Prozedur das Meßprogramm "example.exe" aufgerufen und der Meßwert aus der Datei "messwert.dat" gelesen. In der nachfolgenden IF-Abfrage wird überprüft, ob sich der Wert innerhalb der vorgegeben Toleranzgrenzen befindet. Ist dies der Fall, so wird die Prozedur durch Setzen der Variablen "fehlercode" auf "0" verlassen und mit der Prüfung fortgefahren. Trat ein Fehler auf, so wird die entsprechende Fehlerroutine aufgerufen und der ganze Anweisungsblock, bestehend aus Messung und Toleranzüberprüfung, solange ausgeführt, bis entweder das Ergebnis richtig ist oder der Bediener die Prozedur durch Wahl der Optionen "Abbruch" bzw. "Ignorieren" verlassen will.

3.5.2.1.10. Prozedur Pruefung

Sie stellt den Hauptteil des gesamten Prüfprogramms dar. In ihr werden die gewünschten Spannungen am Prüfling eingestellt und die Prozeduren zur Durchführung einer Messung aufgerufen. Ein Struktogramm befindet sich auf der nächsten Seite.
Am Anfang werden wieder die Portbausteine auf der PC-Erweiterungskarte programmiert. Dann wird aus der Variablen "pruefname", der im Unit "Laden.pas" der Name und Pfad der gewünschten Prüfdatei zugeordnet wurde, die Sachnummer der Baugruppe ermittelt und der Variablen "ausgabename" zugeordnet. Anschließend wird die Bildschirmmaske erstellt und ein Reset des WÖTRA durchgeführt.



Ports programmieren, Sachnummer der Baugruppe aus Dateinamen ermitteln
Pruefschritt fuer Pruefbeginn eingeben
Dateizeiger auf gewuenschten Pruefschritt zeigen lassen
bis Ende Pruefdatei
Datensatz lesen
Messknoten < 999 ?
ja
nein
normaler Pruefdurchgang
Warnung und Kommentar ausgeben
Ermitteln der zu schaltenden Spannungen
Bediener warnen und halten
mehr als eine Spannung?
ja
nein
Relais-Bus umschalten
von Spannung 1 bis Spannung 4
gewuenschte Spannung ermitteln
Spannung auf Relais-Bus schalten
von Koppelpunkt 1 bis Koppelpunkt 8
Koppelpunkt entsprechend gewuenschter Spannung schalten
Messart ?
Durchgangspruefung
Nicht-Durchgangspruefung
Messgeraet
24 V Pruefspannung auf Relais-Bus D0,D1
Messknoten schalten
Messwert ueber A/D-Wandler einlesen
24 V Pruefspannung auf Relais-Bus D0,D1
Messknoten schalten
Messwert ueber A/D-Wandler einlesen
Sollwert,Toleranz ausgeben
Messknoten schalten
Messwert lesen (example.exe)
4<Wert<6?
ja
nein
-1<Wert<+1?
ja
nein
Wert in Toleranz?
ja
nein
naechster Pruefschritt
Programm anhalten
Abbrechen Wiederholen oder Ignorieren
naechster Pruefschritt
Programm anhalten
Abbrechen Wiederholen oder Ignorieren
naechster Pruefschritt
Programm anhalten
Abbrechen Wiederholen oder Ignorieren
Pruefdatei schliessen
neuer Pruefdurchgang oder Hauptmenue ?
wiederhole, bis Rueckkehr zum Hauptmenue gewuenscht
Rueckkehr in die Benutzeroberflaeche

Der eigentliche Prüfablauf befindet sich dann in einer REPEAT-Schleife, die solange durchlaufen wird, bis der Bediener am Ende der Baugruppenprüfung den Menüpunkt zur Rückkehr in die übergeordnete Bedieneroberfläche wählt. Ansonsten wird mit der Prüfung erneut begonnen.
In der Schleife wird der Prüfer zuerst gefragt, bei welchem Prüfschritt er beginnen möchte. Dies ist dann sinnvoll, wenn eine Baugruppe bereits zum Teil geprüft wurde und die schon durchgeführten Schritte nicht ein zweites Mal ablaufen sollen. Der Dateizeiger der Prüfdatei wird dann mit "Seek" auf den entsprechenden Wert gesetzt.
Die nachfolgende While-Schleife sorgt dafür, daß die Prüfung Schritt für Schritt solange erfolgt, bis das Dateiende erreicht und damit die Prüfung abgeschlossen ist.
Ein Prüfschritt beginnt damit, daß ein Datensatz von der Prüfdatei gelesen wird. Gleich darauf wird die Variable "datensatz.messknoteneins" auf den Wert 999 abgefragt . Hierin wird ja unterschieden, ob es sich um einen normalen Prüfschritt handelt, oder ob ein Programmstop zur Warnung des Bedieners erfolgen soll. Handelt es sich um einen realen Prüfschritt, so wird zuerst einmal das Warnungs- und Kommentarfenster am Bildschirm beschrieben.
Dann wird in einer FOR-Schleife ermittelt, wieviele Spannungen an den Prüfling angeschaltet werden sollen. Dies geschieht dadurch, daß das Feld "datensatz.spannung" auf das Vorhandensein des Strings " - " oder "Leist" überprüft wird. Nur wenn beide Strings nicht vorhanden sind, handelt es sich um eine Spannung, die über den Relais-Bus geschaltet werden soll. Ergibt die Überprüfung, daß mehr als eine Spannung benötigt wird, so erfolgt auf der Adresse 402 die Ausgabe des Werts 15, was einer Auftrennung des Relais-Bus ab dem Ausgang 301 entspricht. Dies ist aufgrund der in Kapitel 2.4 erläuterten Gründe notwendig.
Als Nächstes werden dann die entsprechenden Spannungen ausgegeben. Hierfür werden die im Feld "datensatz.spannung" stehenden vier Strings ausgewertet und in einer Case-Anweisung dann die zugehörigen Relais auf den Spannungsumschaltplatinen PL4 und PL5 angesprochen.

Zu diesem Zeitpunkt liegen die Spannungen auf dem Relais-Bus an. Um die Einstellvorgänge am WÖTRA zu beenden, müssen dann in einer FOR-Schleife, die von eins bis acht läuft, nur noch die entsprechenden acht Koppelpunkte geschaltet werden, wodurch dann die Spannungen an den Anschlußpunkten des WÖTRA und damit auch an der Baugruppe anliegen.
Wenn die Einstellungen beendet sind, kann mit der Prüfung begonnen werden. Hier müssen drei Fälle unterschieden werden: Durchgangsprüfung, Nichtdurchgangsprüfung oder eine Messung mit dem Digitalmultimeter.
Eine Durchgangsprüfung läuft folgendermaßen ab:
Zuerst wird auf den Relais-Bus auf die Leitungen D0 und D1 die 24V-Meßspannung und auf D4 der A/D-Wandler-Eingang der Portkarte geschaltet. Der erste der Meßknoten, zwischen denen auf Durchgang geprüft werden soll, wird auf die Leitung D0 geschaltet. Der Zweite wird mit der Meßleitung D4 verbunden. Ist nun ein Durchgang zwischen den beiden Ausgängen in der Baugruppe vorhanden, so muß auf der Meßleitung die 24V-Spannung erscheinen. Dies wird dadurch überprüft, daß ein Spannungswert über den A/D-Wandler eingelesen und auf den Bereich von vier bis sechs Volt überprüft wird. Vier bis sechs Volt deshalb, weil wie in Kapitel 3.5.4.1 beschrieben noch ein Potentiometer zur Spannungsbegrenzung zwischengeschaltet ist. Ergibt die Bereichsüberprüfung ein korrektes Ergebnis, so wird der Anweisungsblock verlassen und zum nächsten Prüfschritt übergegangen. Wenn nicht, so wird die Fehlerroutine wie in Kapitel 3.5.2.1.8 beschrieben abgearbeitet.
Eine Nicht-Durchgangsprüfung läuft ganz entsprechend ab, nur mit dem Unterschied, daß der eingelesene Spannungswert auf den Bereich von -1 bis +1 Volt abgeprüft wird. Soll kein Durchgang zwischen den zu testenden Knoten vorhanden sein, so darf ja auch auf der Meßleitung Relais-Bus-D4 keine Spannung anliegen.
Nun zum Anweisungsblock zur Messung eines Wertes unter Zuhilfename des Digitalmultimeters Fluke 8842A. Hier werden zuerst der Sollwert der Messung sowie die Toleranzangaben im Protofollfenster am Monitor ausgegeben (die Werte hier-

für stehen in den Variablen "datensatz.sollwert", "datensatz.toleranzoben" und "datensatz.toleranzunten") Dann wird das Meßgerät durch Ausgabe des Wertes 9 an der Adresse 401 (Meßgeräteumschaltung PL2) auf die Relais-Bus-Leitungen D0 und D1 geschaltet. Die beiden Ausgänge des WÖTRA B, zwischen denen gemessen werden soll, werden ebenfalls auf diese Leitungen aufgeschaltet. Anschließend erfolgt der Aufruf der in Kapitel 3.5.2.1.9 beschriebenen Prozedur "Messung_Durchfuehren", welche auch bereits die Anweisungen für den Fehlerfall enthält. Ist die Variable "fehlercode" nach der Rückkehr aus der Prozedur auf den Wert zwei gesetzt, so bedeutet dies gewünschten Programmabruch, was dann mit der Exit-Anweisung realisiert wird. Trat kein Fehler auf, so wird das Protokollfenster wieder gelöscht und mit der Prüfung fortgefahren.
Ist das Ende der Prüfdatei erreicht, so wird sie geschlossen und noch einmal zur Sicherheit ein Reset der Knoten durchgeführt.
Zum Schluß wird dann der Bediener in einem neu eröffneten Fenster gefragt, ob er mit einer neuen Prüfung beginnen oder ins Hauptmenü zurückkehren will. Entscheidet er sich für eine Rückkehr ins Hauptmenü, so wird ein abschließender kompletter Reset des WÖTRA durchgeführt und das Unit verlassen. Wünscht er einen erneuten Prüfdurchgang, so wird die übergeordnete REPEAT-Schleife nochmals abgearbeitet, was einem Neustart des Prüfablaufprogramms entspricht.

3.5.2.2. Zu MS-DOS

Mit diesem Menüpunkt kann die Benutzeroberfläche verlassen und zum Betriebssystem zurückgekehrt werden. Im Programm wird dies realisiert, indem die Prozedur "Exit" aufgerufen wird.

3.5.3. Das Menü Messen

Die Routinen für das Durchführen einer Einzelmessung bzw. zum Aufstellen einer Meßreihe waren nicht Bestandteil meiner Diplomarbeit und wurden daher nicht implementiert. Werden sie einmal geschrieben, so können sie im Hauptprogramm der Benutzeroberfläche in der Case-Anweisung unter Punkt sieben und acht eingefügt werden. Die entsprechenden Stellen sind im Listing gekennzeichnet.

Eine Realisierung der beiden Programmpunkte hat jedoch nur dann Sinn, wenn eine Treibersoftware für das Meßgerät erhältlich ist, die direkt in Turbo-Pascal eingebunden werden kann. Im Moment wird das Meßprogramm ja noch als externes EXE-File vom Prüfprogramm aufgerufen. Deshalb dauert das Einlesen eines Meßwerts ca. 3-4 Sekunden (siehe Kapitel 3.3.4). Das Aufstellen einer Meßreihe ist also in vernünftigem zeitlichen Rahmen nicht durchführbar.

3.5.4. Das Menü Test

Unter diesem Menüpunkt sind die verschiedenen Testroutinen des WÖTRA B erreichbar. Sie werden im folgenden genauer beschrieben.

3.5.4.1. Selbsttest

Wie der Selbsttest durchgeführt wird, wurde bereits bei der Beschreibung der Selbsttest-Umschalteplatine in Kapitel 2.4.2 beschrieben, so daß sich dieses Kapitel auf die Softwarebeschreibung dazu beschränken kann.

3.5.4.1.1. Der Programmkopf

Er ist wieder ähnlich aufgebaut wie bei den anderen verwendeten Units. Im Interfaceteil wird die Hauptprozedur "Selbst_Test" zur Verwendung vom Menüprogramm "pruefpr4.pas" aus freigegeben. Der Implementationsteil beginnt mit der Einbindung der Units "objunit.pas" und "global.pas" sowie der Standardunits von Turbo-Pascal. Der Inhalt dieser Units wurde bereits an anderer Stelle ausführlich beschrieben.

3.5.4.1.2. Prozedur Selbsttestmaske

In diesem Programmteil erfolgt die Bildschirmgestaltung sowie das Einlesen der vom Programm benötigten Werte von der Tastatur.
Der Aufbau des Bildschirms ist wieder ähnlich dem der bereits beschriebenen Programmteile, d.h. es wird eine Menü- und Statuszeile in der Farbe grau und ein Arbeitsausschnitt in blau erzeugt. Dann erfolgt die Ausgabe der Überschrift und das Zeichnen des Rahmens für die Werteeingabe im Arbeitsfenster.
In diesem Rahmen wird der Bediener anschließend aufgefordert, den Startknoten für den Test einzugeben. Das Lesen

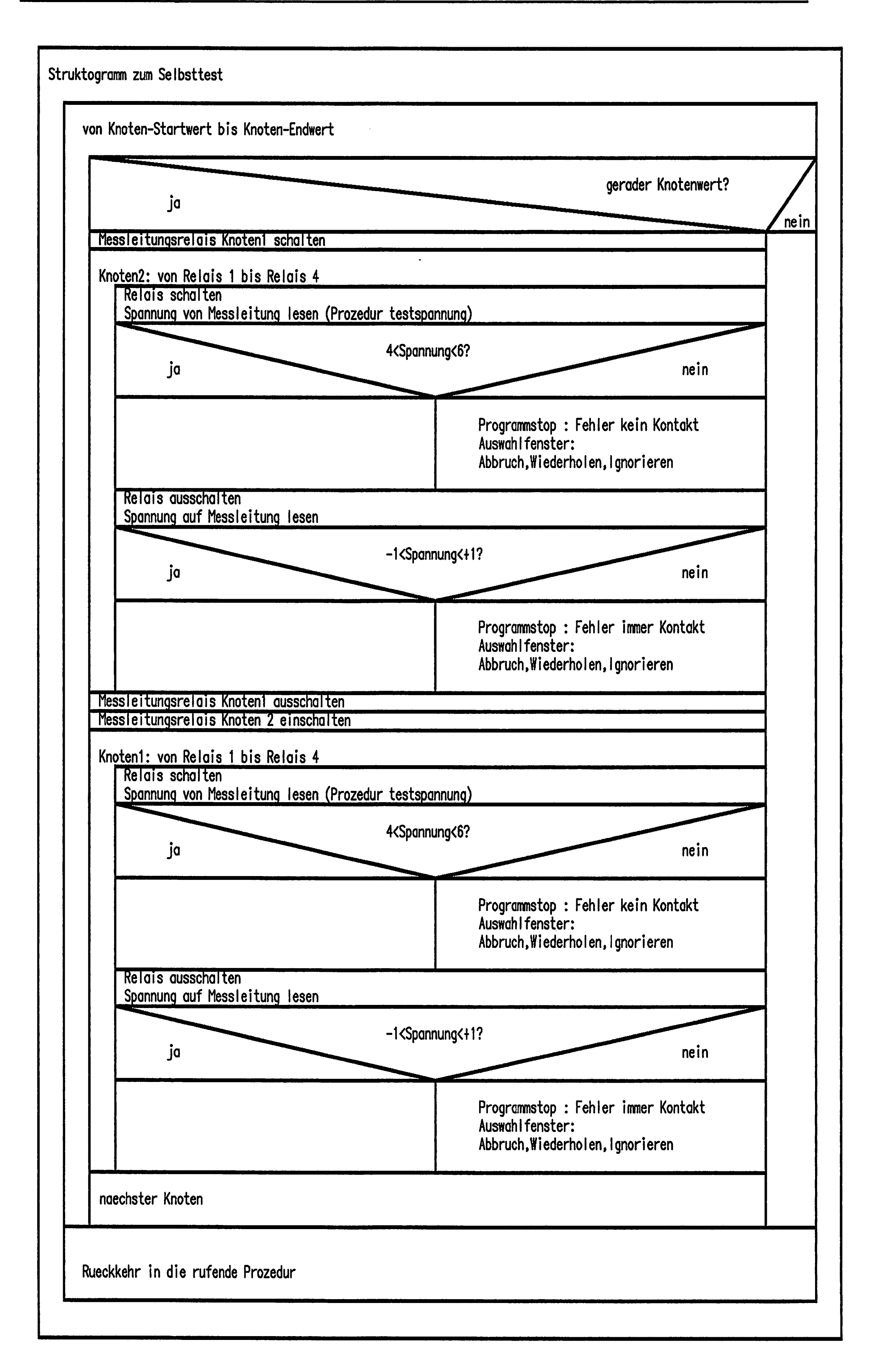
dieses Werts von der Tastatur erfolgt wieder durch die Prozedur "Knoten_Lesen" aus dem Unit "global.pas", da ansonsten bei versehentlichem Drücken der <RETURN>-Taste die vorher aufgebaute Bildschirmmaske teilweise überschrieben werden würde. Gleichzeitig wird durch eine Abfrage verhindert, daß Knotennummern kleiner 1 oder größer 400 eingegeben werden können, was ja nicht sinnvoll wäre. Danach wird bei der Eingabe des Zielknotens für den Selbsttest genauso verfahren.
Da bedingt durch den Aufbau der Kurzschlußstecker für den Selbsttest der Startknoten immer eine ungerade und der Zielknoten eine gerade Zahl sein muß, wird in zwei IF-Abfragen geprüft, ob die eingegebenen Werte dieser Forderung genügen. Ist dies nicht der Fall, so werden sie automatisch vom Programm korrigiert.
Zum Schluß der Prozedur wird dann das Fenster für die Werteeingabe gelöscht und der Bildschirm für die Mitprotokollierung des Testablaufs vorbereitet.

3.5.4.1.3. Funktion Testspannung

Siehe Kapitel 3.5.2.1.5

3.5.4.1.4. Prozedur Test_Ablauf

In dieser Routine werden die Relais zur Durchführung des Selbsttests angesteuert und die vom Durchgangsprüfungseingang der PC-Portkarte (A/D-Wandler-Kanal 9) eingelesenen Spannungspegel ausgewertet. Ein Struktogramm dazu folgt auf der nächsten Seite.
Die ganze Prozedur ist in eine große FOR-Schleife eingebettet, in der die Knotennummern vom Start- bis zum Endwert durchlaufen werden. In der Schleife wird durch eine IF-Abfrage sichergestellt, daß der ganze Testdurchgang immer nur bei jeder zweiten Knotennummer ausgeführt wird. Das ist notwendig, da die Ausgänge des WÖTRA B jedesmal paarweise geprüft werden.


Struktogramm zum Selbsttest
von Knoten-Startwert bis Knoten-Endwert
gerader Knotenwert?
ja
nein
Messleitungsrelais Knoten1 schalten
Knoten2: von Relais 1 bis Relais 4
Relais schalten
Spannung von Messleitung lesen (Prozedur testspannung)
4<Spannung<6?
ja
nein
Programmstop : Fehler kein Kontakt
Auswahlfenster:
Abbruch,Wiederholen,Ignorieren
Relais ausschalten
Spannung auf Messleitung lesen
-1<Spannung<+1?
ja
nein
Programmstop : Fehler immer Kontakt
Auswahlfenster:
Abbruch,Wiederholen,Ignorieren
Messleitungsrelais Knoten1 ausschalten
Messleitungsrelais Knoten 2 einschalten
Knoten1: von Relais 1 bis Relais 4
Relais schalten
Spannung von Messleitung lesen (Prozedur testspannung)
4<Spannung<6?
ja
nein
Programmstop : Fehler kein Kontakt
Auswahlfenster:
Abbruch,Wiederholen,Ignorieren
Relais ausschalten
Spannung auf Messleitung lesen
-1<Spannung<+1?
ja
nein
Programmstop : Fehler immer Kontakt
Auswahlfenster:
Abbruch,Wiederholen,Ignorieren
naechster Knoten
Rueckkehr in die rufende Prozedur

Der eigentliche Test beginnt dann damit, daß die Variable "messpannung", die später den vom A/D-Wandler eingelesenen Wert enthält, auf Null gesetzt wird. Nachfolgend wird beim ersten des zu testenden Knotenpaares das Relais Nummer 5, also das Relais der Meßleitung, eingeschaltet.

Hierauf werden in einer WHILE-Schleife die oberen vier Relais des zweiten Knotens der Reihe nach ein- und ausgeschaltet. Nach jedem Einschaltvorgang wird über die Prozedur "Testspannung" ein Spannungswert über die Meßleitung des Wötra eingelesen. Hat das entsprechende Relais korrekt angezogen, so muß auf dieser Leitung eine Spannung von 24V anliegen. Die dazugehörige Abfrage bezieht sich jedoch auf den Bereich von vier bis sechs Volt, da die Spannung zur Durchgangsprüfung durch ein nachgeschaltetes Potentiometer auf 5V begrenzt wird. Dies wurde gemacht, da der A/D-Wandler nicht für eine Spannung von 24V ausgelegt ist. Liegt also die eingelesene Spannung im richtigen Bereich, so kann mit der Prüfung fortgefahren werden. Ist dies nicht der Fall, so wird am Bildschirm ein Fenster geöffnet, in dem die Fehlerart sowie die betroffenen Knoten und Relais angezeigt werden. Der Prüfer wird dann vor die Wahl gestellt, den Testlauf abzubrechen, den Testschritt zu wiederholen oder den Fehler zu ignorieren, was eine Fortsetzung des Selbsttests zur Folge hat.

Ist die Prüfung des Einschaltvorganges des Relais abgeschlossen, so wird es wieder ausgeschaltet. Jetzt erfolgt ein Test daraufhin, ob es "klebt", d.h. ob es trotz Ausschaltens noch Kontakt gibt. Dazu wird über die Prozedur "Testspannung" wieder ein Wert von der Meßleitung eingelesen. Dieser Wert wird daraufhin überprüft, ob er im Bereich von -1 bis +1 Volt liegt. Dies müßte der Fall sein, wenn das Relais ordnungsgemäß abgefallen ist. Tritt hier jedoch ein Fehler auf, so wird wieder ein Fenster am Bildschirm geöffnet, in dem Fehlerart und Fehlerort angezeigt werden. Der Bediener hat wieder die Wahl zwischen Abbrechen, Wiederholen und Ignorieren.

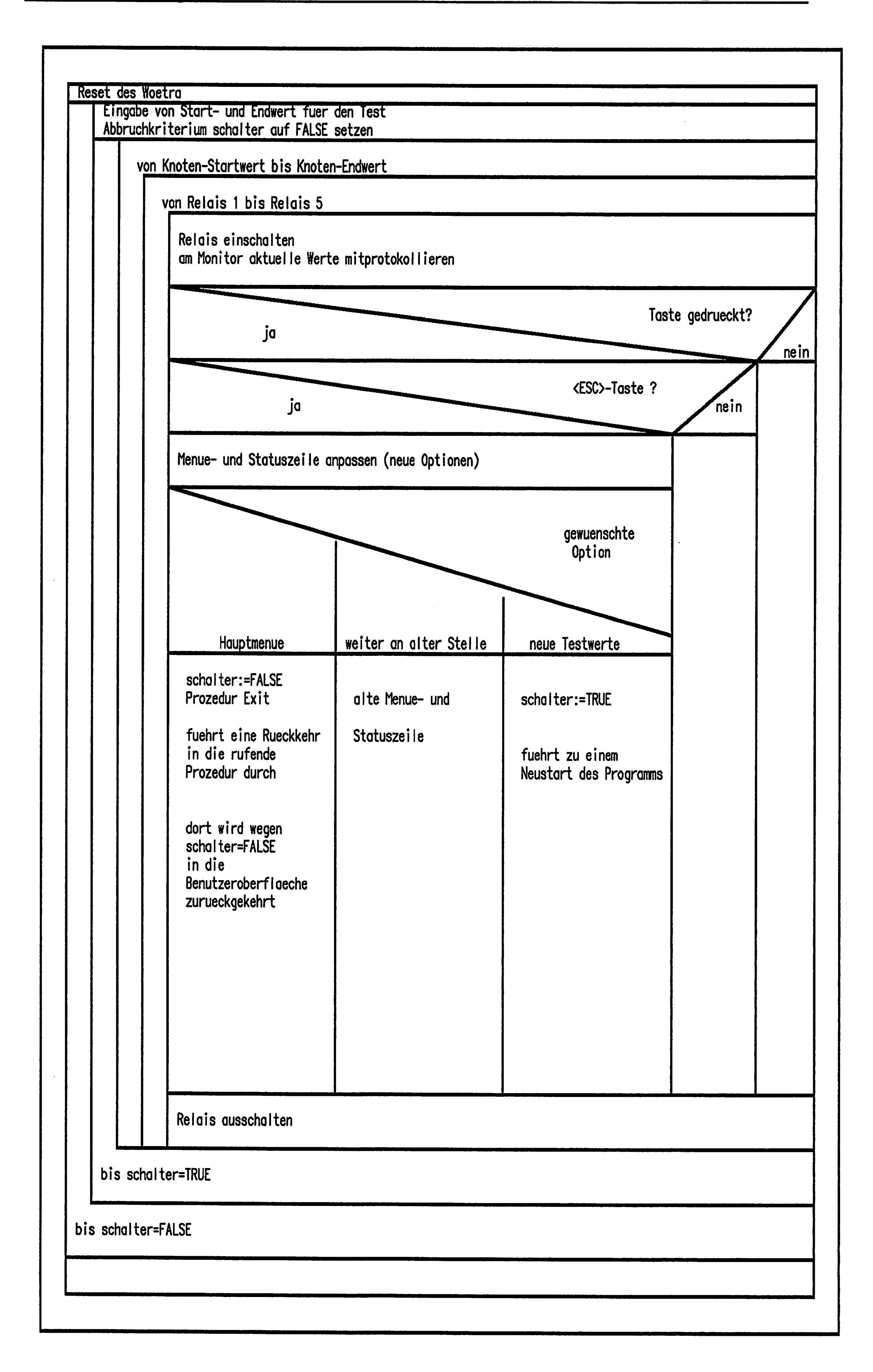
Trat bei beiden Tests des Relais kein Fehler auf, so wird mit dem Nächsten in gleicher Weise verfahren.
Der Ablauf ist dann für alle vier Relais des zweiten Knotens der gleiche.
Nach diesen Durchläufen sind dann das Relais Nummer 5 des ersten der beiden Knoten und die vier oberen Relais des zweiten geprüft.
Jetzt wird genau umgekehrt vorgegangen: das Relais Nummer 5 des zweiten Knotens wird eingeschaltet und nacheinander die vier oberen Relais des ersten. Die Prüfung, ob sie Kontakt geben oder nicht, geschieht wieder in genau der gleichen Weise wie bereits beschrieben. Auch die Fehlerbehandlung wird entsprechend abgearbeitet. Treten auch hier keine unzulässigen Zustände der Relais auf, so werden die gesetzten Relais des aktuellen Knotenpaares gelöscht und zum Nächsten übergegangen. Dies erfolgt solange, bis der Endknoten des zu prüfenden Bereichs erreicht ist.

3.5.4.1.5. Prozedur Selbst_Test

Diese Prozedur wird vom Menüprogramm bei Auswahl des Menüpunkts "Selbsttest" aufgerufen. Es programmiert den Baustein 8255 der PC-Portkarte, führt zur Sicherheit einen Relais-Reset durch und bereitet in der Prozedur "Adport" aus dem Unit "global.pas" den A/D-Wandler-Port auf das Einlesen von Spannungswerten vor. Anschließend werden auf der Selbsttestplatine die für den Test benötigten Relais eingeschaltet und auf der Meßgeräteumschaltung die Meßart Durchgangsprüfung gewählt.
Dann wird die vorher beschriebene Prozedur "Test_Ablauf" aufgerufen. Ergaben sich keine Fehler, oder wurden die Fehler alle ignoriert, so wird eine Erfolgsmeldung ausgegeben und der Benutzer aufgefordert, eine beliebige Taste zur Rückkehr ins Hauptmenü zu drücken.

3.5.4.2. Dauertest

Beim Dauertest des WÖTRA B werden die Relais von einem Startknoten bis zu einem Zielknoten nacheinander einzeln angesteuert. Diese Art des Tests war besonders in der Aufbauphase wichtig, da mit seiner Hilfe überprüft werden konnte, ob die Kabel der Verbindungen zwischen den Dekodierplatinen und den Relaiskarten in der richtigen Reihenfolge gesteckt waren. Doch auch in der Betriebsphase hat diese Testart seine Berechtigung, und zwar dann, wenn z.B. eine defekte Dekoder- oder Relaisplatine repariert bzw. ausgetauscht wird und anschließend eine Funktionsprüfung erfolgen soll.
Der Programmkopf hat dieselbe Struktur wie der unter "Selbsttest" beschriebene, d.h. im Interfaceteil wird die Testprozedur zur Einbindung in andere Programme freigegeben und der Implementationsteil beginnt mit der Einbindung der nun schon mehrfach beschriebenen Units "global" und "objunit", gefolgt vom Variablenteil. Das folgende Bild zeigt das Struktogramm zur Durchführung des Dauertests. Es bezieht sich nicht auf eine bestimmte Prozedur des Units "dauertes.pas", sondern soll den Menüpunkt "Dauertest" als Ganzes verdeutlichen.


Reset des Woetra
Eingabe von Start- und Endwert fuer den Test
Abbruchkriterium schalter auf FALSE setzen
von Knoten-Startwert bis Knoten-Endwert
von Relais 1 bis Relais 5
Relais einschalten
am Monitor aktuelle Werte mitprotokollieren
Taste gedrueckt?
ja
nein
<ESC>-Taste ?
ja
nein
Menue- und Statuszeile anpassen (neue Optionen)
gewuenschte Option
Hauptmenue
weiter an alter Stelle
neue Testwerte
schalter:=FALSE
Prozedur Exit
fuehrt eine Rueckkehr in die rufende Prozedur durch
dort wird wegen schalter=FALSE in die Benutzeroberflaeche zurueckgekehrt
alte Menue- und Statuszeile
schalter:=TRUE
fuehrt zu einem Neustart des Programms
Relais ausschalten
bis schalter=TRUE
bis schalter=FALSE

3.5.4.2.1. Prozedur Dauertestmaske

Der Bildschirm soll während des Dauertestlaufs wieder den gleichen Aufbau erhalten, wie in allen anderen Programmteilen. Deshalb wird zu Beginn der Prozedur wieder Menü- und Statuszeile sowie der Arbeitsbildschirm erzeugt und farblich gestaltet. Dann wird die Überschrift ausgegeben und der Rahmen für die Eingabeaufforderung an den Bediener gezeichnet. Als Nächstes werden die für den Test benötigten Werte, also Start- und Zielknoten, von der Tastatur eingelesen. Für jeden der Werte erfolgt sofort nach der Eingabe eine Bereichsüberprüfung (die Knotennummern müssen zwischen 1 und 400 liegen). Ergibt sich kein Fehler, so wird das Programm fortgesetzt, andernfalls wird eine REPEAT-Schleife solange durchlaufen, bis die Eingabe korrekt ist.
Sind beide Werte richtig ermittelt worden, so wird am Bildschirm ein Fenster geöffnet, in dem beim nachfolgenden Testlauf die aktuell angesprochenen Ausgänge und Relais mitprotokolliert werden.

3.5.4.2.2. Prozedur Test_Ablauf

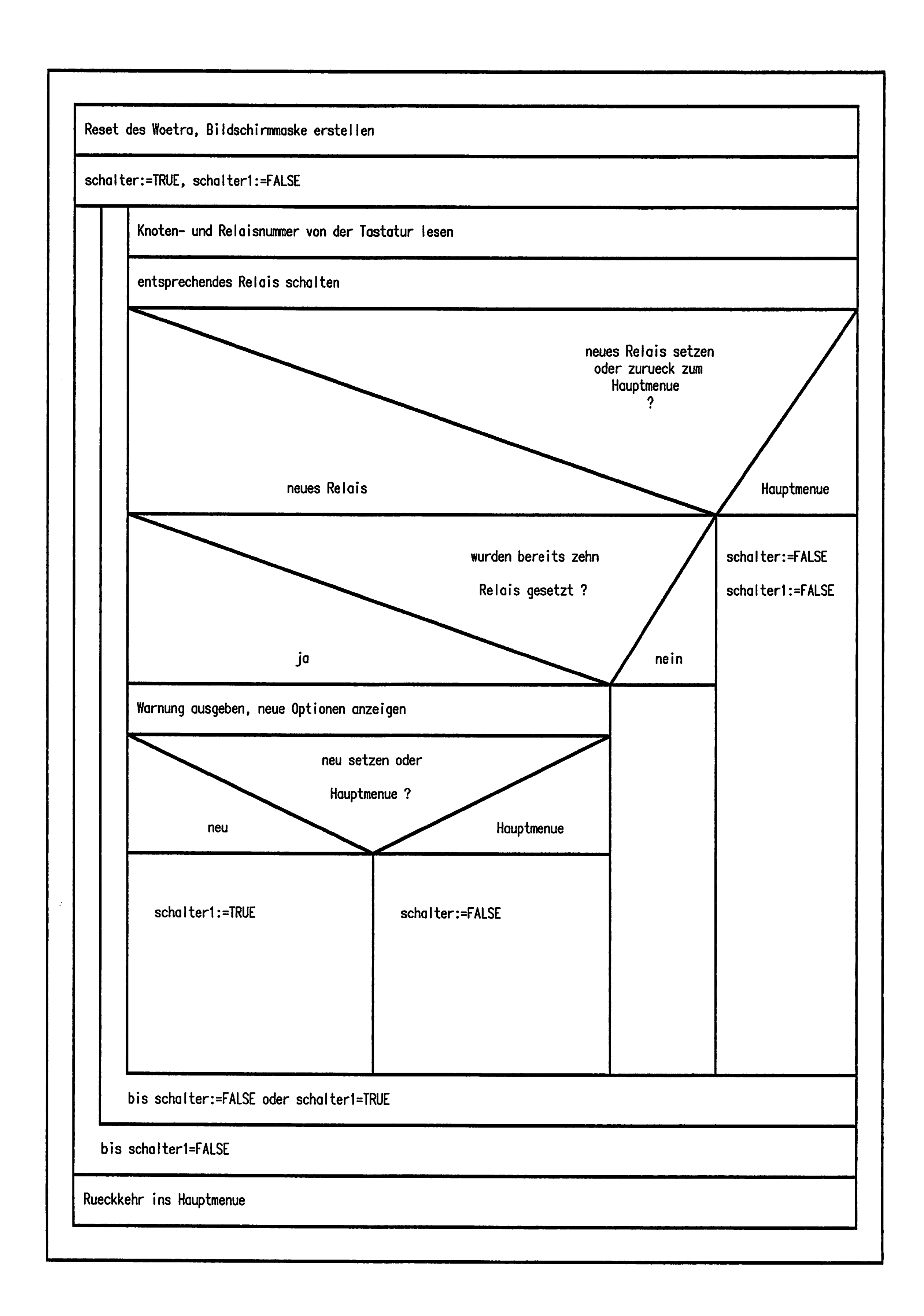
Durch diese Prozedur werden die Relais angesteuert sowie die Handhabung der Menü- und Statuszeile geregelt. Dazu wird während des Tests in der Statuszeile, also am unteren Bildschirmrand die Meldung ausgegeben, daß der Programmlauf mit <ESC> unterbrochen werden kann. Wird dies getan, so soll die Meldung wieder verschwinden und statt dessen in der Menüzeile die Anzeige der verschiedenen Optionen zur Fortsetzung des Programms erscheinen. Dies sind Rückkehr zum Hauptmenü, also der Benutzeroberfläche, Neustart des Tests mit Eingabe von Start- und Endknoten oder Weiterführen des Tests an der Stelle, an der er unterbrochen wurde.
Zur Ansteuerung der Relais sind zwei Schleifen vorgesehen, eine FOR-Schleife, um die Knotennummern durchlaufen zu lassen und eine darin eingebettete WHILE-Schleife, welche die fünf Relais pro Knoten nacheinander anspricht. Die jeweili-

gen Werte werden dann von der Prozedur "Ausgeben" zum WÖTRA B geschickt und damit die Relais geschaltet. Hierauf werden die zugehörigen Nummern am Bildschirm ausgegeben, wodurch es dem Bediener ermöglicht wird, den Test in allen Einzelheiten zu verfolgen.
Im nächsten Programmschritt wird abgefragt, ob die <ESC>-Taste gedrückt wurde. Ist dies der Fall, so wird die Meldung in der Statuszeile gelöscht und die Menüzeile mit dem Angebot der erreichbaren Programmoptionen beschrieben. Je nachdem, welche Taste dann gedrückt wird, werden in einer Case-Anweisung die entsprechenden Programmteile angewählt. Soll zum Hauptmenü zurückgekehrt werden, wird die Variable "schalter", welche in der Prozedur "Dauer_Test" in einer REPEAT-Schleife abgefragt wird, auf FALSE gesetzt und die Prozedur mit "Exit" verlassen. Wird die Option "Weiter" gewünscht, so stellt das Programm den ursprünglichen Zustand der Menü- und Statuszeile wieder her und setzt den Programmlauf fort. Um einen Neustart des Programms zu erhalten, wird der Variablen "schalter" der Wert TRUE zugewiesen, wodurch vermieden wird, daß in "Dauer_Test" die Prozedur beendet wird.
Ist der beschriebene Anweisungsblock abgearbeitet, so löscht das Programm das zuletzt gesetzte Relais wieder und geht zum nächsten über.

3.5.4.2.3. Prozedur Dauer_Test

Sie hat nur die Aufgabe, den Ausgangsbaustein der Portkarte zu programmieren und in einer Schleife den Test solange durchlaufen zu lassen, bis vom Bediener nach dem Drücken der <ESC>-Taste die Menüoption zur Rückkehr ins Hauptmenü gewählt wird. Erreicht wird dies, indem die Variable "schalter" in einer REPEAT-Schleife abgefragt wird. Ein Programmabruch erfolgt nur dann, wenn diese Variable auf FALSE gesetzt wurde.

3.5.4.3. Einzeltest



Der Menüpunkt Einzeltest ermöglicht es, bis zu 10 beliebige Relais des WÖTRA B einzuschalten. Es ist aber pro Knoten immer nur ein Relais schaltbar. Dies dient der Vermeidung von Kurzschlüssen auf dem Relais-Bus.
Der Einzeltest kann dazu hergenommen werden, bei einem Fehler auf einer der Relaiskarten das defekte Relais zu lokalisieren, indem alle Relais des betroffenen Bereichs der Reihe nach einzeln angesteuert werden.
Das obige Struktogramm soll die Gesamtarbeitsweise des Units "einzelte.pas" verdeutlichen. Es nimmt nicht Bezug auf eine bestimmte der anschließend im einzelnen beschriebenen Prozeduren.

3.5.4.3.1. Prozedur Maske

Wie bei den anderen Testprogrammen wird beim Einzeltest der Bildschirm in Menü- und Statuszeile sowie Arbeitsbildschirm aufgeteilt. Dies geschieht in der vorliegenden Prozedur.
Ferner werden noch die Rahmen für die Eingabe der Werte für Knoten- und Relaisnummer sowie für die Mitprotokollierung der bereits gesetzten Relais am Monitor erstellt.

3.5.4.3.2. Prozedur Eingabe

Soll ein Relais angesteuert werden, so ist die Eingabe der Knotennummer, auf der es sich befindet, sowie seiner Stelle innerhalb des Knotens erforderlich. Das Einlesen der Werte hierfür erfolgt wieder über die Prozedur "Knoten_Lesen" aus dem Unit "global.pas", wodurch vermieden wird, daß bei einer versehentlichen Falscheingabe die Bildschirmmaske zerstört oder das Programm abgebrochen wird. Für die Knotennummer erfolgt noch eine Überprüfung, ob sie im zulässigen Bereich zwischen 1 und 400 liegt. Für die Relaisnummer wird auf den Bereich von 1 bis 5 geprüft. Sind beide Werte über Tastatur eingelesen, wird die Prozedur beendet. Treten Fehler auf, so werden die entsprechenden Eingabeschleifen solange durchlaufen, bis der Bediener sinnvolle Knoten- und Relaisnummern eingibt.

3.5.4.3.3. Prozedur Ausgabe

Mit dem Programm Einzeltest können bis zu 10 Relais angesteuert werden. Um die Übersicht über die bereits aktivierten Relais nicht zu verlieren, werden sie am Bildschirm unter Angabe von Knoten- und Relaisnummer angezeigt. Dies geschieht in der Prozedur "Ausgabe". Ferner wird in der Menüzeile eine Auswahlmöglichkeit für den Prüfer zwischen Rückkehr zum Hauptmenü und erneutem Setzen eines Relais angezeigt.

3.5.4.3.4. Prozedur Auswahl

Die vorher angesprochene Auswahlmöglichkeit wird in dieser Prozedur ausgewertet. Entscheidet sich der Prüfer zu einer Rückkehr in die Benutzeroberfläche, so wird in der Case-Anweisung unter Punkt 72 die Prozedur mit "Exit" verlassen, nachdem die Variable "schalter", die ein Abbruchkriterium für die Hauptprozedur "Einzel_Test" ist, auf FALSE gesetzt wurde.
Soll das Programm nicht verlassen, sondern mit dem setzen eines Relais fortgeführt werden, so wird der Variablen "schalter" der Wert TRUE zugewiesen, womit vermieden wird, daß das Programm an der oben angedeuteten Stelle beendet wird.

3.5.4.3.5. Prozedur Test_Ablauf

Sie führt das eigentliche Schalten der Relais aus. Je nach Relaisnummer, die gesetzt werden soll, wird der Variablen "daten" ein Wert zwischen 1 und 16 zugewiesen und unter der gewünschten Adresse ausgegeben.

3.5.4.3.6. Prozedur Einzel_Test

Sie ist die Hauptprozedur des Units "einzelte.pas" und wird vom Benutzerprogramm "pruefpr4.pas" bei der Auswahl des Menüpunkts "Einzeltest" aufgerufen.
Sie beginnt wie alle Prüfprogramme des WÖTRA B mit der Programmierung der PC-Portkarte. Anschließend wird die Bildschirmmaske erstellt und ein Reset durchgeführt. Dann wird die Variable "k", die als Zähler für die bereits gesetzten Relais hergenommen wird, auf Null gesetzt. Hierauf erfolgt die Eingabe der gewünschten Werte von der Tastatur und das Schalten des entsprechenden Relais mit der Prozedur "Ausgabe". Dann wird dem Benutzer die Auswahl zwischen Rückkehr zum Hauptmenü und Fortfahren des Tests ermöglicht. Wählt er "Fortfahren", so wird zuerst die Variable "k" und damit die Anzahl der bereits gesetzten Relais überprüft. Hat sie den Wert 10 erreicht, so ergeht eine Warnung an den Bediener, daß nur maximal 10 Relais gesetzt werden können. Bestätigt er die Warnung mit <RETURN>, so wird er vor die Wahl gestellt, entweder das Programm zu verlassen, oder einen Neustart durchzuführen.
Wählt er Neustart, so wird die Variable "schalter1", die als Abbruchkriterium für die Prozedur "Einzel_Test" dient, auf TRUE gesetzt und damit die Programmschleife erneut durchlaufen.
Hat er sich für eine Rückkehr zum Hauptmenü entschlossen, so wird die gesamte Prozedur mit "Exit" verlassen.

3.5.4.4. Spannungen

Dieser Programmteil war nicht mehr Bestandteil meiner Diplomarbeit. Wenn die dafür vorgesehenen Routinen einmal realisiert sind, können sie im Hauptprogramm der Benutzeroberfläche in der Case-Anweisung unter Punkt 12 eingefügt werden. Das Programmlisting enthält an der entsprechenden Stelle einen Hinweis.

4. Bedienungsanleitung zur WÖTRA B Software

4.1. Installation

Um eine Installation der WÖTRA-Software durchzuführen, muß folgendermaßen vorgegangen werden:
Zuerst muß sichergestellt sein, daß sich folgende Verzeichnisse auf der Festplatte des Steuerechners befinden:

- "C:\TP5\" Hier muß der Turbo-Pascal-Compiler mit all seinen Standardunits installiert werden.
- "C:\MSPASCAL\" Dieses Verzeichnis muß den MS-Pascal 4.0 Compiler der Firma Microsoft enthalten.

Als nächstes müssen folgende Dateien von der WÖTRA B Quelltextdiskette in das Verzeichnis "C:\TP5\" kopiert werden:
pruefpr4.pas, mausunit.pas, global.pas, objunit.pas, fenster.pas, filename.pas, director.pas, windo.pas, helpme.pas, laden.pas, startpru.pas, selbstte.pas, dauertes.pas, einzelte.pas
Ist dies geschehen, so wird Turbo-Pascal aufgerufen und das File "pruefpr4.pas" eingeladen. Im Menü "Compile" wird die Option "Destination Disk" ausgewählt und mit "Build" die Compilierung gestartet. Als Ergebnis erhält man das ausführbare Programm "pruefpr4.exe".
Damit dieses ordnungsgemäß funktionieren kann, müssen von der WÖTRA B Programmdiskette noch die Dateien "diwö.exe" in das Verzeichnis "C:\TP5\" und "example.exe" in das Verzeichnis "C:\MSPASCAL\" kopiert werden.
Zum Abschluß der Installation wird noch die Batch-Datei "WOETRA.BAT" von der Programmdiskette in das Root-Verzeichnis der Festplatte geladen.
Die WÖTRA-Software kann dann durch Eingabe von "WOETRA" gestartet werden.

4.2. Die Programmbedienung

4.2.1. Die Benutzeroberfläche

Wie bereits gesagt, wird die WÖTRA-Software durch Eingabe von "WOETRA" aufgerufen. Es erscheint dann folgendes Bild am Monitor:

Prüfdatei Aktionen Messen Test

Prüfprogramm WÖTRA B

Copyright 1989 Wolfgang Schraml

Bitte Taste drücken

F1-Hilfe F10-Menü

Es stellt die in Kapitel 3.4.10 beschriebene Eingangsmeldung dar. Nun muß eine beliebige Taste gedrückt werden, worauf der Bildschirm folgendermaßen aussieht:

Prüfdatei Aktionen Messen Test

F1-Hilfe F10-Menü

An dieser Stelle muß einer der Menüpunkte ausgewählt werden. Dies kann auf drei verschiedene Art und Weisen geschehen:

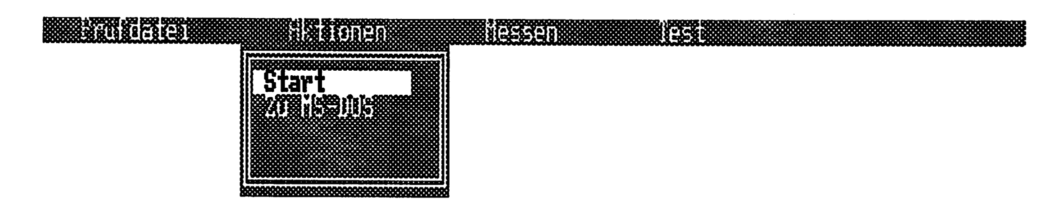
- Drücken des Anfangsbuchstabens des gewünschten Menüs
- Anklicken des Namens in der Menüzeile mit der Maus
- Drücken von <F10>

Ist man sich einmal unklar über die Bedienung , so kann mit <F1> eine kurze Online-Hilfe angefordert werden.
Hat man dann eine der drei Möglichkeiten gewählt, so befindet man sich im entsprechenden Untermenüfenster. Hierbei sind folgende vier Fälle möglich:

Menü "Prüfdatei"



Menü "Aktionen"

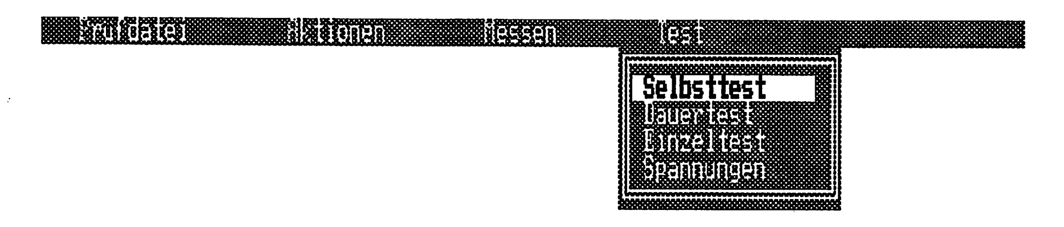


Menü "Messen"



F1-Hilfe F10-Menü

Menü "Test"



F1-Hilfe F10-Menü

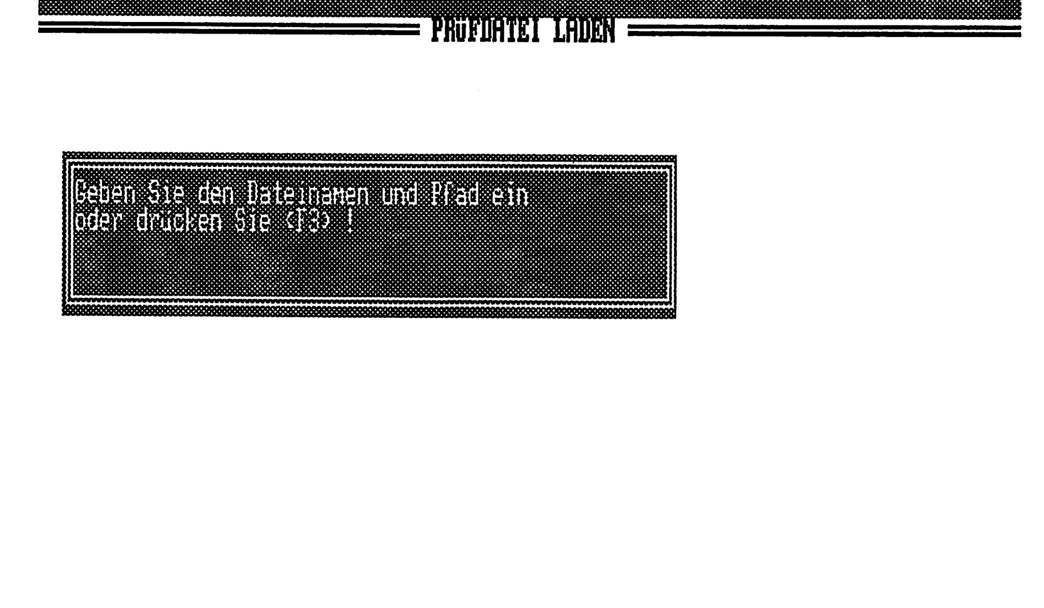
In jedem der einzelnen Fenster kann man sich mittels der Maus oder der Cursor-Steuertasten beliebig bewegen. Eine Auswahl des aktuellen Menüpunktes erfolgt durch Drücken des linken Mausknopfes oder der <RETURN>-Taste.
Hierauf wird dann der Bildschirm gelöscht und je nach gewähltem Menüpunkt neu gestaltet. Wie er im einzelnen aussieht und welche Eingaben zu machen sind, folgt in den nächsten Kapiteln.

4.2.2. Die Menüpunkte

In diesem Kapitel wird die Bedienung der Menüpunkte "Prüfdatei Laden", "Aktionen Start", "Aktionen Zu MS-DOS", "Test Selbsttest", "Test Dauertest" und "Test Einzeltest" beschrieben, da sie von mir bearbeitet wurden. Der Menüpunkt "Prüfdatei erstellen" ist in der Diplomarbeit von Markus Kölbl ausführlich beschrieben. Die anderen Untermenüs sind noch nicht implementiert.

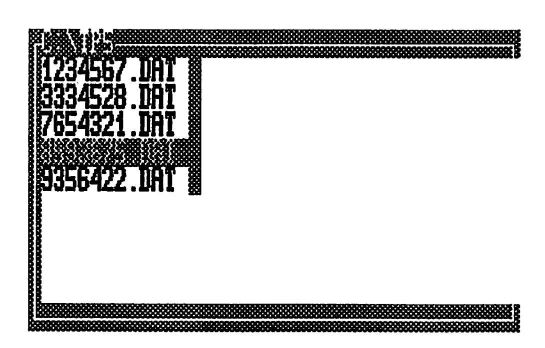
4.2.2.1. Prüfdatei Laden

Wird dieser Punkt angewählt, erscheint folgende Meldung am Monitor:



Sind Pfad und Dateinamen der gewünschten Prüfdatei bekannt, so können diese direkt eingegeben und mit <RETURN> bestätigt werden. Wenn nicht, so kann durch Drücken der Funktionstaste <F3> eine Übersicht über alle vorhandenen Prüfdateien angefordert werden. Dies zeigt folgendes Bild:

PRÜFDATEI LADEN



Mit den Cursor-Steuertasten kann man sich in der Übersicht frei bewegen. Ist der Markierungsbalken auf die gewünschte Datei plaziert, so wird sie mit <RETURN> ausgewählt.
Nach der Auswahl wird die Datei noch einmal entsprechend dem nächsten Bild angezeigt, und dem Bediener die Möglichkeit zu geben, bei eventueller Falschauswahl Korrekturen vornehmen zu können.

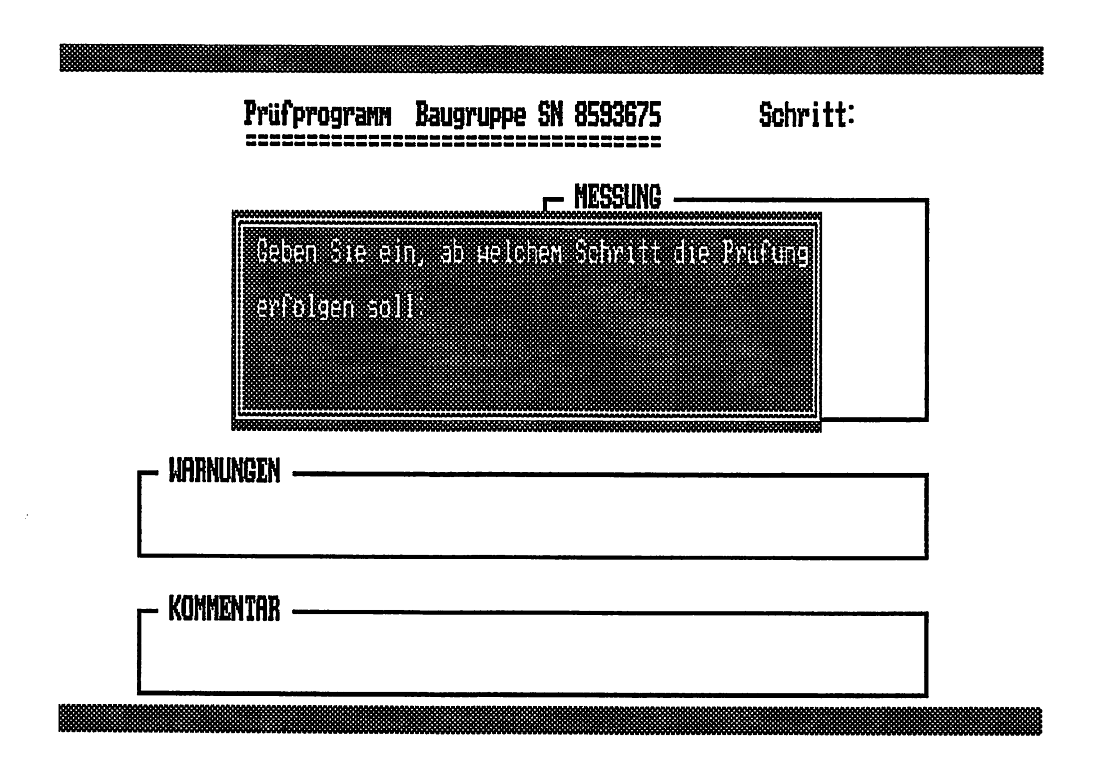


Sie haben folgende Datei ausgewahlt :
C:\TP5\8593675.DAT
Einverstanden ? (j/n)

Wird mit <j> bestätigt, so wird zum Hauptmenü zurückgekehrt. Wird <n> eingegeben, so beginnt der Dateiauswahlzyklus von Neuem.

4.2.2.2. Aktionen Start

Unter diesem Menüpunkt findet die eigentliche Baugruppenprüfung statt. Vor dem Aufruf muß jedoch eine Prüfdatei gemäß Kapitel 4.2.2.1. geladen worden sein.
Es erscheint folgende Meldung des Programms:



Hier kann ein beliebiger Prüfschritt als Startwert gewählt werden. Wird eine Baugruppe zum ersten Mal getestet, so muß hier natürlich die "1" eingegeben werden.
Wird mit <RETURN> bestätigt, so läuft das Prüfprogramm automatisch ab. Nur in den folgenden zwei Fällen stoppt das Programm.

1.Fall: Es wurde ein Fehler in der Baugruppe erkannt. Tritt dies ein, so wird der Bediener folgendermaßen darauf hingewiesen:

Prüfprogramm Baugruppe SN 8593675 Schritt: 2
==================================

Fehler
Durchgang zwischen 102 und 133

Abbruch --->A
Wiederholen --->W
Ignorieren --->I

NICHTDURCHGANGSPRÜFUNG

Sollwert:
Istwert :

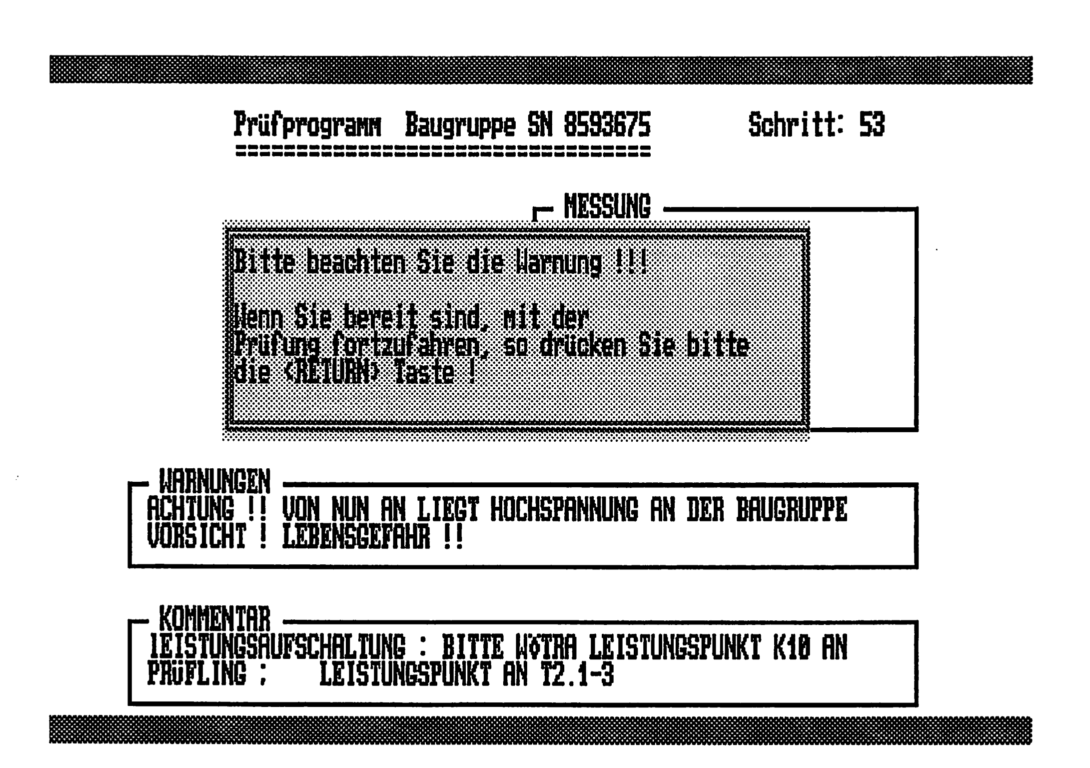
Toleranz:
oben:
unten:

WARNUNGEN

KOMMENTAR
S6
D711X2.6-U5-K2.2

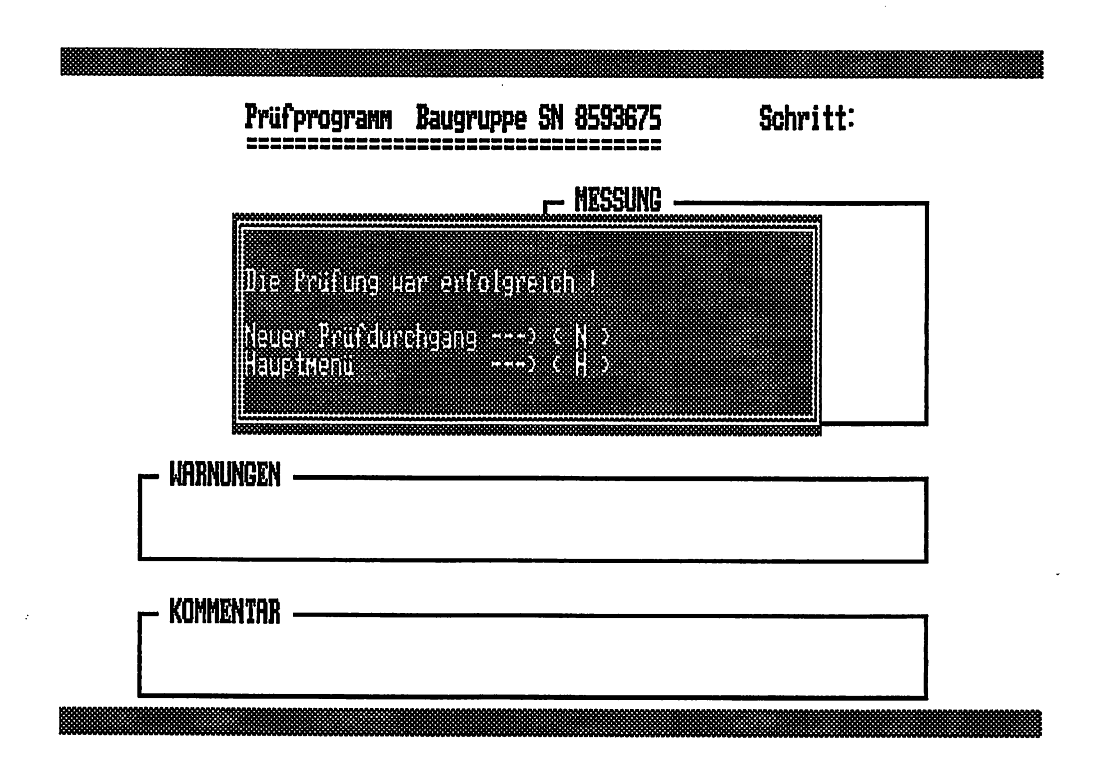
Das Bild zeigt die Fehlermeldung bei einer Nicht-Durchgangsprüfung. Bei den anderen Meßarten lauten sie dann entsprechend etwas anders. Das Prinzip der Fehlerbehandlung bleibt jedoch stets das gleiche.
Entscheidet man sich für <Abbruch>, so wird die Prüfung beendet und ins Hauptmenü zurückgekehrt. Wählt man <Wiederholen>, so wird der zuletzt durchgeführte Prüfschritt wiederholt. Tritt der Fehler erneut auf, so erfolgt wieder ein Programmstop. Ist er jedoch behoben worden, setzt das Programm die Prüfung fort. Wurde <Ignorieren> eingegeben, so beachtet das Programm den Fehler nicht weiter und beginnt mit dem nächsten Prüfschritt.

2.Fall: Der Bediener soll vor der weiteren Programmausführung eine Warnung erhalten. Ein Beispiel für diesen Fall zeigt das nächste Bild:



Es handelt sich um die Warnung des Bedieners vor der Hochspannung, die während der nachfolgenden Prüfschritte an der Baugruppe anliegt. Diese Warnungen sollten also keinesfalls auf die leichte Schulter genommen werden.
Wird nach dem Lesen der Meldungen mit <RETURN> bestätigt, so setzt das Programm den Test fort.

Sind dann schließlich alle Prüfschritte bearbeitet worden, so wird mittels folgender Meldung gefragt, ob noch ein weiterer Prüfdurchgang gewünscht wird, oder ob ins Hauptmenü verzweigt werden soll.



Wird <N> eingegeben, so beginnt das Programm von vorne. Drückt man jedoch die Taste <H>, so wird in die Benutzeroberfläche zurückgekehrt.

4.2.2.3. Aktionen Zu MS-DOS

Will man das Programm verlassen, so muß diese Option gewählt werden. Der Bildschirm wird gelöscht und das DOS-Prompt erscheint wieder auf dem Monitor.

4.2.2.4. Test Selbsttest

Dieser Menüpunkt führt den in Kapitel 3.5.4.1. beschriebenen Selbsttest durch. Die folgende Eingangsmeldung erscheint nach dem Aufruf am Bildschirm:

SELBSTTEST

Test soll erfolgen :
von Knoten : 1
bis Knoten : 16

Hier müssen vom Bediener Start- und Endwert (zwischen 1 und 400) für den Test angegeben werden. Wurde beides korrekt eingegeben, so läuft der Test ab, wobei der aktuelle Testschritt wie auf der nächsten Seite dargestellt mitprotokolliert wird:

SELBSTTEST

Test soll erfolgen :
von Knoten : 1
bis Knoten : 16

Testablauf :
Knoten1: 1 Relaisnummer: 1
Knoten2: 2 Relaisnummer: 5

Tritt ein Fehler auf, so erfolgt ein Programmstop. Je nachdem, um welche Art Fehler es sich handelt, wird eine Meldung entsprechend dem nächsten Bild am Monitor ausgegeben. Die Wirkung der zur Verfügung stehenden Programmoptionen ist dieselbe wie in Kapitel 4.2.2.2 beschrieben.

SELBSTTEST

Test soll erfolgen :
von Knoten : 1
bis Knoten : 16

Testablauf :
Knoten1: 1 Relaisnumm
Knoten2: 2 Relaisnumm

Achtung!! Knoten 1 Relais 1
kein Kontakt bei ein

Abbruch ---> A
Ignorieren ---> I
Wiederholen ---> W

4.2.2.5. Test Dauertest

Die Eingangsmeldung zur Durchführung des Dauertests ist dieselbe wie unter "Selbsttest" beschrieben. Deshalb wird hier auf eine erneute Wiedergabe verzichtet.
Nach der Eingabe von Start- und Endwert (zwischen 1 und 400) wird der Test unter folgender Bildschirmausgabe durchgeführt:

DAUERTEST

Test soll erfolgen :
von Knoten : 1
bis Knoten : 16

Testablauf :
Knotennummer:3
Relaisnummer:5

<ESC> - Abbruch

Im unteren Fenster wird jeweils das gerade aktivierte Relais mit der zugehörigen Knotennummer angezeigt.

Ein Abbruch des Tests ist durch Drücken von <ESC> jederzeit möglich. Es erfolgt dann eine Rückkehr ins Hauptmenü.

4.2.2.6. *Test Einzeltest*

Unter diesem Menüpunkt können bis zu zehn Relais des WÖTRA einzeln angesprochen werden. Die Eingabe der entsprechenden Knoten- und Relaisnummer muß in der folgenden Maske geschehen:

Hauptmenü Weiter

EINZELTEST

1. Relais soll gesetzt werden :
Knotennummer : 1
Relaisnummer : 1

Testablauf :
Knotennummer: 1
Relaisnummer: 1

Dabei sind Knotennummern von 1 bis 400 und Relaisnummern von 1 bis 5 zulässig. Im unteren Bildschirmfenster wird mitprotokolliert, welche Relais bereits gesetzt wurden.
Nach jeder Eingabe hat der Bediener die Auswahl zwischen Rückkehr zum Hauptmenü und Fortsetzten des Tests.
Wurde das zehnte Relais gesetzt, so erscheint nachfolgende Warnung:

EINZELTEST

10. Relais soll gesetzt werden :
Knotennummer : 10

Es dürfen nur maximal 10 Relais gesetzt werden!

weiter --> RETURN

Relaisnummer: 1 2 3 4 5 1 2 3 4 5

Bestätigt man mit <RETURN>, so erscheint in der Menüzeile die Anzeige der zur Verfügung stehenden Möglichkeiten zur Fortsetztung des Programms. Je nachdem, welche Option gewählt wird, erfolgt eine Rückkehr ins Hauptmenü oder ein erneuter Ablauf des Einzeltests. Dies zeigt das letzte Bild dieses Kapitels.

Hauptmenü Neu setzen

EINZELTEST

10. Relais soll gesetzt werden :
Knotennummer : 10
Relaisnummer : 5

Testablauf :

Knotennummer:	1	2	3	4	5	6	7	8	9	10
Relaisnummer:	1	2	3	4	5	1	2	3	4	5

5. Ausblick

Nach der Beendigung einer Arbeit sollte immer die selbstkritische Betrachtung des Geleisteten stehen. Darunter soll nicht verstanden werden, daß das Ergebnis der Arbeit einer Abwertung unterzogen werden soll. Vielmehr sollte man sich Gedanken darüber machen, welche Verbesserungen in Zukunft noch möglich wären.
In meinem Fall wären noch folgende Erweiterungen für die Zukunft in Erwägung zu ziehen:

- Ergänzung der von mir noch nicht implementierten Untermenüpunkte (siehe entsprechende Stellen in Kapitel 3).

- Erweiterung des gesamten Baugruppenprüfgeräts auf acht Relais-Bus-Leitungen (dadurch wäre es dann nicht mehr nötig, ab Adresse 300 eine Relais-Bus-Umschaltung durchzuführen). Diese Neuerung wird im Ansatz bereits von zwei Praktikanten entwickelt. Hier fließen auch schon die Erfahrungen, die ich während des Aufbaus des Prüfgeräts gemacht habe, mit ein (als Hauptpunkt ist die Frage der Verbindung zwischen Dekoder- und Relaisplatinen zu nennen. Die 96-poligen Wrap-Stecker, die in der jetzigen Version Verwendung finden, sind äußerst empfindlich in der Handhabung. Sie sollen durch stabilere Steckverbinder ersetzt werden).

- Umschreiben der Steuersoftware für das IEC-Bus gesteuerte Meßgerät auf die Programmiersprache "C". Dadurch wäre es besser möglich, eventuell auftretende Meßgerätefehler abzufangen und eine Zerstörung der Bildschirmmaske des Prüfprogramms zu verhindern.

- Überarbeitung der Fehlermeldungen des Prüfprogramms. Sie könnten so gestaltet werden, daß Warnungen an den Bediener von den Fehlermeldungen farblich unterschieden werden. Bei Warnungen vor einer Leistungsaufschaltung an die Baugruppe

könnte auch noch ein akustisches Signal mit einbezogen werden.

Einen großen Komplex innerhalb von Erweiterungen könnte die Implementierung einer Routine zur Selbstprogrammierung des WÖTRA B einnehmen. Hierunter ist zu verstehen, daß eine als fehlerfrei bekannte Baugruppe angeschlossen wird, und die Prüfdateierstellung automatisch abläuft. Dies wäre jedoch nur für reine Durchgangsprüfungen ohne erheblichen Mehraufwand möglich. Dazu müßte die Baugruppe an den WÖTRA adaptiert werden, und programmgesteuert jeder Ausgang zu jedem anderen Ausgang auf Verbindung geprüft werden. Festgestellte Verbindungen müßten in einer Prüfdatei entsprechend dem Kapitel 3.5.2.1.1 abgespeichert werden. Die für die einzelnen Prüfschritte benötigten Kommentare und Hinweise an den Prüfer sowie eventuell nötige Messungen an der Baugruppe müßten natürlich weiterhin von Hand erstellt werden.

Ein solch komplexes Prüfgerät wie der WÖTRA B und seine Steuersoftware kann wohl nie als völlig vollendet gelten. Es werden in der Praxis immer wieder unvorhergesehene Betriebszustände auftreten, die Änderungen nach sich ziehen. In der jetzigen Phase ist jedoch ein solider Grundstein geschaffen worden, auf den sich alle zukünftigen Erweiterungen beziehen können.

6. Schlußwort

Die Diplomarbeit stellt für jeden Studenten den Abschluß seines Studiums dar. Hier muß er zeigen, ob er das theoretische Wissen, daß er sich in den Vorlesungen und Praktika bzw. den Hausübungen angeeignet hat, auch in die Praxis umsetzen kann.
Aus diesem Grund finde ich es auf jeden Fall empfehlenswert, wenn die Diplomarbeit in einem Betrieb und nicht an der Fachhochschule bearbeitet wird. Man wird hier direkt mit den Problemen der Praxis konfrontiert. Diese müssen auch möglichst schnell gelöst werden, da für einen Betrieb vergeudete Zeit sofort einen nicht unerheblichen Kostenfaktor darstellt.
Der Diplomand erhält also einen direkten Einblick in die bevorstehende berufliche Zukunft. Er lernt, sich in einer Gruppe zu bewähren und zu behaupten, anstehende Arbeiten zu koordinieren und mit den Kollegen abzusprechen sowie die eigene Arbeit durch gezielte Planung effektiver zu gestalten.
Abschließend möchte ich sagen, daß ich die Erfahrungen, die ich während meiner Diplomarbeit bei der Siemens AG in Kemnath gemacht habe, nicht missen möchte. Ich kann nur jedem angehenden Studienabsolventen raten, sich um eine Diplomandenstelle in einem Betrieb zu bewerben. Er wird es nicht bereuen.

7. Anhang

7.1. Stücklisten

Selbsttest und Leistungspunkt

K1-K6	Siemens Kleinschaltrelais N V23016-B006-A101
D1-D6	Diode 1N4007
	Stiftleiste 31-polig
	Lochrasterplatine Europa-Format

Relaisbus-Umschaltung

K1-K4	Siemens Kleinschaltrelais N V23016-B006-A101
D1-D4	Diode 1N4007
	Stiftleiste 31-polig
	Lochrasterplatine Europa-Format

Meßgeräteumschaltung

K1-K5	Siemens Kleinschaltrelais E2V23037-B005-A101
D1-D5	Diode 1N4007
	Stiftleiste 31-polig
	Lochrasterplatine Europa-Format

Spannungsumschaltung

K1-K10	Siemens Kleinschaltrelais E2V23037-B005-A101
D1-D10	Diode 1N4007
	Stiftleiste 31-polig
	Lochrasterplatine Europa-Format

Versuchsschaltung Spannungsüberwachung

IC1	Spannungsregler 7815
IC2	" 7915
IC3	U/I-Wandler 1B21 (Analog Devices)
IC4	OPV LM 1458N
R1,R2	Metallschichtwiderstand 27K
R3	Spindeltrimmpoti 500 Ohm
C1,C2	Elko 1000u/40V
C3,C4	0,1u/40V
C5,C6	Elko 4,7u/40V
D1	Brückengleichrichter B80C1500
T1	Printtrafo 220V/2*16V 12,5VA
	Lochrasterplatine Europa-Format

7.2. Technische Daten

Baugruppenprüfgerät WÖTRA B

Steuerrechner	Siemens PCD-2
Anzahl der Koppelpunkte	400
Prüfspannungen	220V, 110V, 24V AC
	24V, 0-100V DC
Meßarten	Durchgangsprüfung
	Nicht-Durchgangsprüfung
	Widerstandsmessung
	Gleich-, Wechselspannung
	Gleich-, Wechselstrom
gleichzeitig anlegbare Spg.	4
Netzanschluß	220V AC

7.3. Verzeichnis der verwendeten Literatur

Turbo-Pascal 5.5
Benutzerhandbuch, Referenzhandbuch
Heimsoeth und Borland, München 1988/1989

Microsoft Pascal 4.0
Reference Manual, User's Guide, Code View/Editor
Microsoft Corporation, 1987

Fluke 8842A Instruction Manual
John Fluke Mfg.Co., Inc, 1985

Philips Instrument Drivers PM2232
User Manual Pascal Language
Philips Gloeilampenfabrieken Eindhofen, 1987

Handbuch zur Erweiterungskarte me-30 der Firma Impec

Siemens Datenbuch Relais 1984/84

Linear Products Databook
Data Conversion Products Databook
Analog Devices 1988

MS-DOS für Siemens PCD-2
Benutzerhandbuch
Microsoft Corporation, 1986

PC Intern 2.0 Systemprogrammierung
Michael Tischer, Data Becker Düsseldorf, 1989

Arbeiten mit Microsoft WORD 5.0
Microsoft Corporation, 1989

Das große Buch zu WORD 5.0
Data Becker, Düsseldorf 1989

Microsoft Macro Assembler 5.0
Handbücher
Microsoft Corporation, 1987

MaNager, Turbo-Pascal 4.0 unterstützt die Maus
c't Magazin für Computertechnik 1988, Heft 10/S.196
Verlag Heinz Heise, Hannover

Turbo-Fenster auf dem PC
Schnelle Windows unter Turbo-Pascal
c't Magazin für Computertechnik 1987, Heft 6/S.114

Directorys unter Turbo-Pascal
Die Turbo-Pascal-Werkstatt
DOS International 1989 Heft 6/S.158

Diplomarbeit Stefan Seitz 1989
Fachhochschule Regensburg

Diplomarbeit Markus Kölbl 1989
Fachhochschule Nürnberg

7.4. Die Quelltexte der Programme

7.4.1. mausunit.pas

```
UNIT mausunit ;

{ enthält die Funktionen zur Maussteuerung.    }
{ wird von Pruefpr4.pas verwendet.             }

INTERFACE

USES Crt,Dos ;

PROCEDURE Zeige_Maus ;
     { Mauscursor wird am Bildschirm sichtbar }
PROCEDURE Verstecke_Maus ;
     { Mauscursor verschwindet vom Bildschirm }
PROCEDURE Maus_Position (VAR x,y:WORD) ;
     { ermittelt die Mausposition }
PROCEDURE Setze_Maus (x,y:WORD) ;
     { setzt Mauscursor auf x,y   }
PROCEDURE Reset_Maus ;
     { Mausparameter in Grundzustand }
PROCEDURE Aendere_Verhaeltnis(hori,verti:WORD);
     { verlangsamt Mausbewegung }
FUNCTION Knopf_Links:BOOLEAN ;
     { ermittelt Status der linken Maustaste }
FUNCTION Knopf_Rechts:BOOLEAN ;
     { ermittelt Status der rechten Maustaste}
FUNCTION Maus_Im_Bereich(x1,y1,x2,y2:WORD):BOOLEAN ;
   { überprüft, ob sich die Maus im angegebenen Ausschnitt befindet }

IMPLEMENTATION

CONST
  Rax:WORD=0 ;
  Rbx:WORD=0 ;
  Rcx:WORD=0 ;
  Rdx:WORD=0 ;
  Rsi:WORD=0 ;
  Rdi:WORD=0 ;

{--------------------------------------------------------------}

FUNCTION BitTest(TestByte,BitNummer:BYTE):BOOLEAN ;
   { ermittelt, ob das Bit mit der Nummer Bitnummer von }
   { TestByte größer oder kleiner Null ist.             }

BEGIN
  TestByte:=TestByte AND (1 shl BitNummer) ;
  BitTest:=(TestByte > 0);
END; { von BitTest }

{--------------------------------------------------------------}

PROCEDURE Maus;
```

```
   { führt den Maus-Interrupt-Aufruf durch. }

  VAR regs:REGISTERS;
  BEGIN
    regs.ax:=Rax ;
    regs.bx:=Rbx ;
    regs.cx:=Rcx ;
    regs.dx:=Rdx ;
    regs.si:=Rsi ;
    regs.di:=Rdi ;
    Intr(51,regs);
    Rax:=regs.ax ;
    Rbx:=regs.bx ;
    Rcx:=regs.cx ;
    Rdx:=regs.dx ;
  END; { von Maus }

{-------------------------------------------------------------}

PROCEDURE Zeige_Maus ;
   { macht den Mauscursor am Bildschirm sichtbar }

BEGIN
  Rax:=1 ;    { Funktionsnummer in AX-Register }
  Maus ;      { Interrupt }
END;  { von Zeige_Maus }

{-------------------------------------------------------------}

PROCEDURE Verstecke_Maus ;
   { macht den Mauscursor am Bildschirm unsichtbar }

BEGIN
  Rax:=2 ;     { Funktionsnummer in AX-Register }
  Maus ;       { Interrupt }
END;  { von Verstecke_Maus }

{-------------------------------------------------------------}

PROCEDURE Maus_Position (VAR x,y:WORD) ;
   { liefert in x und y die Koordinaten des Mauscursors zurück }

BEGIN
  Rax:=3 ;     { Funktionsnummer in AX-Register }
  Maus;        { Interrupt }
  x:=Rcx ;     { x-Koordinate steht im CX-Register }
  y:=Rdx ;     { y-Koordinate steht im DX-Register }
END;  { von Maus_Position }

{-------------------------------------------------------------}

PROCEDURE Setze_Maus (x,y:WORD) ;
   { bringt den Mauscursor an die mit x und y gekennzeichnete }
   { Stelle }

BEGIN
  Rax:=4 ;    { Funktionsnummer in AX-Register }
  Rcx:=x ;    { x-Koordinate in CX-Register    }
```

```
  Rdx:=y ;    { y-Koordinate in DX-Register    }
  Maus;       { Interrupt }
END;  { von Setze_Maus }

{-----------------------------------------------------------------}

PROCEDURE Reset_Maus ;
   { bringt Maus in den Grundzustand; Mauscursor sichtbar und }
   { in Bildschirmmitte                                       }

BEGIN
  Rax:=0;    { Funktionsnummer in AX-Register }
  Maus;      { Interrupt }
END;  { von Reset_Maus }

{-----------------------------------------------------------------}

FUNCTION Knopf_Links:BOOLEAN ;
  { liefert den Wert TRUE zurück, wenn linke Maustaste gedrückt }
  { andernfalls FALSE }

BEGIN
  Rax:=3 ;                    { Funktionsnummer in AX-Register }
  Maus ;                      { Interrupt }
  IF Bittest(Rbx,0) THEN Knopf_Links:=TRUE    { Taste gedrückt }
     ELSE Knopf_Links:=FALSE ;                { Taste nicht gedrückt }
END;  { von Knopf_Links }

{-----------------------------------------------------------------}

FUNCTION Knopf_Rechts:BOOLEAN ;
 { liefert den Wert TRUE zurück, wenn rechte Maustaste gedrückt }
 { andernfalls FALSE }

BEGIN
  Rax:=3 ;                   { Funktionsnummer in AX-Register }
  Maus ;                     { Interrupt }
  IF BitTest(Rbx,1) THEN Knopf_Rechts:=TRUE  { Taste gedrückt }
     ELSE Knopf_Rechts:=FALSE ;              { Taste nicht gedrückt }
END;   { von Knopf_Rechts }

{-----------------------------------------------------------------}

FUNCTION Maus_Im_Bereich(x1,y1,x2,y2:WORD):BOOLEAN ;
  { liefert den Wert TRUE zurück, wenn sich die Maus in dem }
  { durch x1,y1,x2,y2 angegebenen Ausschnitt befindet       }

VAR x,y:WORD ;
BEGIN
  IF x2<x1 THEN    { falls x2<x1, die beiden Werte tauschen }
    BEGIN
      x:=x1;
      x1:=x2;
      x2:=x;
    END;
  IF y2<y1 THEN    { falls y2<y1, die beiden Werte tauschen }
    BEGIN
      y:=y1;
```

```
      y1:=y2;
      y2:=y;
    END;
  Maus_Position(x,y);          { Mauskoordinaten ermitteln }
  Maus_Im_Bereich:=( ((x>=x1) AND (x<=x2-1)) AND ((y>=y1) AND (y<=y2-
1)) );
                               { Bereichsüberprüfung }
END;   { von Maus_Im_Bereich }

{----------------------------------------------------------------}

PROCEDURE Aendere_Verhaeltnis(hori,verti:WORD);
  { verändert die Geschwindigkeit der Mausbewegung }

BEGIN
  Rax:=15;          { Funktionsnummer in AX-Register }
  Rcx:=hori;        { Horizontal }
  Rdx:=verti;       { Vertikal   }
  Maus;             { Interrupt  }
END;  { von Aendere_Verhaeltnis }

END.   { von mausunit.pas }
```

7.4.2. global.pas

```
UNIT Global ;
  { enthält Datendefinitionen und Prozeduren, welche von }
  { mehreren anderen Units verwendet werden.             }

INTERFACE

TYPE einzelschritt=RECORD
                     koppelpunkt:ARRAY[1..8] OF INTEGER ;
                     sollwert,toleranzoben,toleranzunten:REAL ;
                     spannung:ARRAY[1..5] OF STRING ;

pruefschritt,messknoteneins,messknotenzwei:INTEGER;
                     kommentar,warnung,messart:STRING;
                   END;
           { Diese Record enthält einen einzelnen Prüfschritt.     }
           { Jede Wötra-Prüfdatei ist aus solchen Einzelschritten  }
           { aufgebaut.                                            }


VAR  pruefname:STRING[64];  { Pfad und Dateiname der Prüfdatei.     }
                            { wird in Laden.pas der gewünschte Wert }
                            { zugewiesen. }
     keychar:CHAR;          { Puffervariable Zeichenlesen }
                            { von der Tastatur. }

PROCEDURE Ausgeben (adresse:WORD;daten:BYTE);
PROCEDURE Port_Programmieren ;
     { Programmierung der Ausgabe-Ports }
PROCEDURE Adport;
     { Programmierung des A/D-Wandler-Ports(8255) }
FUNCTION SpannungLesen(kanal:INTEGER):INTEGER;
PROCEDURE Relais_Reset;
     { setzt Relais der Koppelpunkte zurück }
PROCEDURE Umschaltung_Reset;
     { setzt Relais der Umschaltplatinen zurück }
PROCEDURE Cursor_Off ;
     { schaltet den Cursor aus     }
PROCEDURE Knoten_Lesen(var knoten:INTEGER);
     { liest 3-stellige Knotennummer  }
     { Unterdrückung der Return-Taste }


{-----------------------------------------------------------------}

IMPLEMENTATION
USES Crt,Dos;


PROCEDURE Ausgeben (adresse:WORD;daten:BYTE);
         { gibt über Port a,b und c die Adressen und Daten aus }

  VAR
    aport,bport,cport:BYTE;         { Port a,b,c }
    schleife        :INTEGER;       { Zählervariable }
  BEGIN
```

```
    aport:=adresse AND $FF;
    bport:=(adresse SHR 8) AND $FF;
    cport:=daten;
    PORT[$70A]:=cport;            { Daten }
    FOR schleife:=1 TO 400 DO;    { Verzögerung für OptoKoppler}
    PORT[$708]:=aport;            { Low-Byte der Adresse }
    PORT[$709]:=bport OR $80;     { High-Byte der Adresse }
    FOR schleife:=1 TO 400 DO;    { Verzögerung für Optokoppler }
    PORT[$709]:=bport AND $7F;    { WR-Signal (A15) Low }
    FOR schleife:=1 TO 400 DO;    { Verzögerung für Optokoppler }
    PORT[$709]:=bport OR $80;     { WR-Signal (A15) high }
  END;       { von Ausgeben }

{-----------------------------------------------------------------}

PROCEDURE Port_Programmieren ;
    { Programmierung des Ports(8255) }

  BEGIN
    PORT[$70B]:=$80;      { Ausgabe des Steuerbyte }
  END;  { von Port_Programmieren }

{-----------------------------------------------------------------}

PROCEDURE Adport;
     { Programmierung des A/D-Wandler-Ports(8255) }

  BEGIN
    PORT[$703]:=$92;      { Ausgabe des Steuerbyte }
  END;  { von Adport }

{---------------------------------------------------------------}

FUNCTION SpannungLesen(kanal:INTEGER):INTEGER;
  { liest über den A/D-Wandler-Kanal "kanal" }
  { einen Spannungswert ein                  }

VAR i:INTEGER;
  BEGIN
    Port[$702]:=02;
    Port[$702]:=kanal*$10+3;
    FOR i:=1 TO 6 DO;
    SpannungLesen:=((Port[$701] AND $0F) SHL 8)+Port[$700];
  END;  { von SpannungLesen }

{---------------------------------------------------------------}

PROCEDURE Relais_Reset;
    { läßt alle Relais des Wötra abfallen }

VAR adresse:WORD ;
  BEGIN
      adresse:=00;             { Anfangsadresse}
      REPEAT
        Ausgeben(adresse,0); { alle Relais löschen }
        adresse:=adresse+1;  { nächste Adresse }
      UNTIL adresse=410;     { bis alle Adressen }
  END;  { von Relais_Reset }
```

```
{--------------------------------------------------------------------}

PROCEDURE Umschaltung_Reset;
     { läßt alle Relais der Umschaltplatinen abfallen }

VAR adresse:WORD ;
  BEGIN
    adresse:=399;      { Anfangsadresse }
     REPEAT
       Ausgeben(adresse,0);  { Relais löschen }
       adresse:=adresse+1;   { nächste Adresse }
     UNTIL adresse=410;      { bis alle Adressen }
  END;  { von Umschaltung_Reset }

{--------------------------------------------------------------------}

PROCEDURE Cursor_Off ;
    { schaltet den Cursor aus     }
VAR  register:REGISTERS ;
  BEGIN
    register.bx:=0;
    register.dx:=6657;
    register.ax:=512;
    Intr(16,register);
  END;  { von Cursor_Off }

{--------------------------------------------------------------------}

PROCEDURE Knoten_Lesen(var knoten:INTEGER);
  { Lesen einer dreistelligen Knotennummer von Tastatur. }
  VAR zahlenfolge:STRING;
      keychar:CHAR;
      zaehler,code:INTEGER;
BEGIN
  zahlenfolge:='';
    REPEAT                {solange Tastatur einlesen bis Ziffer}
      keychar:=ReadKey;                         {eingegeben}
    UNTIL (Ord(keychar) > 47) AND (Ord(keychar) < 58);
  zahlenfolge:=keychar;          { Eingabe abspeichern }
  Write(keychar);                { Echo am Bildschirm  }
  zaehler:=1;                    { Zähler in Grundzustand }
  REPEAT                         { bis Enter gedrückt oder 3 Stellen}
      keychar:=ReadKey;
      IF (Ord(keychar)>47) AND (Ord(keychar)<58) THEN
                                 { wenn gültiges Zeichen }
        BEGIN
          Write(keychar);        { Echo am Bildschirm }
          zahlenfolge:=zahlenfolge+keychar;
          zaehler:=zaehler+1;
        END;
  UNTIL (zaehler=3) or (Ord(keychar)=13);
                          { bis Enter oder drei Stellen }
  Val(zahlenfolge,knoten,code);            { String ---> Integer }
END;     { von Knoten_Lesen }


END. { von global.pas }
```

7.4.3. objunit.pas

```
UNIT objunit ;
    { enthält Funktionen zur Rahmenzeichnung }
INTERFACE

USES Crt;

TYPE  fenster=object                        {Objekt für Rahmenzeichnung}
        x1,y1,x2,y2:INTEGER;                        {Rahmenkoordinaten}
        PROCEDURE Init(Ix1,Iy1,Ix2,Iy2 :INTEGER);
        PROCEDURE Zeigen;
        PROCEDURE Loeschen;
      END;


IMPLEMENTATION


PROCEDURE fenster.Init(Ix1,Iy1,Ix2,Iy2 :INTEGER);
{Übergibt die gewünschten Fensterkoordinaten an }
{das Objekt Fenster                             }
  BEGIN
    x1:=Ix1;Y1:=Iy1;x2:=Ix2;y2:=Iy2;
  END; {fenster.Init}

{-------------------------------------------------------------------}

PROCEDURE fenster.Zeigen ;
{zeichnet das gewünschte Fenster    }
VAR i:INTEGER ;
  BEGIN
    GotoXY(x1,y1);Write(#218);     {Eckpunkte zeichnen}
    GotoXY(x2,y1);Write(#191);
    GotoXY(x1,y2);Write(#192);
    GotoXY(x2,y2);Write(#217);
    FOR i:=(x1+1) TO (x2-1) DO     {oberer und unterer Rand}
      BEGIN
        GotoXY(i,y1);
        Write(#196);
        GotoXY(i,y2);
        Write(#196);
      END;
    FOR i:=(y1+1) TO (y2-1) DO     {linker und rechter Rand}
      BEGIN
        GotoXY(x1,i);
        Write(#179);
        GotoXY(x2,i);
        Write(#179);
      END;
  END;     {fenster.zeigen}

{------------------------------------------------------------------}

PROCEDURE fenster.Loeschen ;
{löscht das angegebene Fenster }
VAR i:INTEGER ;
```

```
  BEGIN
    GotoXY(x1,y1);Write(' ');       {Eckpunkte löschen}
    GotoXY(x2,y1);Write(' ');
    GotoXY(x1,y2);Write(' ');
    GotoXY(x2,y2);Write(' ');
    FOR i:=x1 TO x2 DO              {oberen und unteren Rand löschen}
      BEGIN
        GotoXY(i,y1);
        Write(' ');
        GotoXY(i,y2);
        Write(' ');
      END;
    FOR i:=y1 TO y2 DO              {linken und rechten Rand löschen}
      BEGIN
        GotoXY(x1,i);
        Write(' ');
        GotoXY(x2,i);
        Write(' ');
      END;
  END;    {fenster.löschen}

END. { von objunit.pas }
```

7.4.4. example.pas

```
PROGRAM example(INPUT, OUTPUT);

{ führt die Messung über den IEC-Bus durch. Verwendet }
{ das Include-File Drivers.inc der Philips-Treiber-   }
{ Software. }
{ wurde mit dem MS-Pascal-Compiler 4.0 compiliert.    }

CONST DRIVERS = TRUE;
{$include: 'DRIVERS.INC'}

VAR
  messwert,ergebnis  : REAL8;
  name,ausdruck:STRING(23);
  i,k,l:INTEGER;

{-------------------------------------------------------------------}

PROCEDURE Messartlesen ;
     { liest aus dem File paramet.dat die Art der Messung, }
     { die durchgeführt werden soll                        }

  VAR f :FILE OF STRING(23) ;

  BEGIN
    Assign(f,'c:\mspascal\paramet.dat'); { zuordnen des Suchpfades }
    Reset(f);                            { zum Lesen Öffnen        }
    Read(f,name);                        { Messart lesen           }
    Close(f);                            { Datei schließen         }
  END;      {von Messartlesen}

{-------------------------------------------------------------------}

PROCEDURE Messwertschreiben (messwert:REAL8) ;
     { schreibt den vom Digitalmultimeter übernommenen }
     { in die Datei messwert.dat                       }

  VAR f:FILE OF REAL8 ;

  BEGIN
    Assign(f,'c:\mspascal\messwert.dat');{ zuordnen des Suchpfades }
    Rewrite(f);                          { zum Schreiben Öffnen    }
    Write(f,messwert);                   {Meßwert in Datei schreiben}
    Close(f);                            { Datei schließen }
  END; {von Messwertschreiben}


{********************** HAUPTPROGRAMM ******************************}

BEGIN
  drivers_initialization; { Rücksetzen der IEC-Bus-Karte }
  Messartlesen ;          { Lesen der gewünschten Meßart }
  ausdruck:=name ;
  IF ausdruck='GLEICHSPANNUNGSMESSUNG ' THEN set_function (DMM1,
VOLT_DC)
```

```
    ELSE  IF ausdruck='WECHSELSPANNUNGSMESSUNG' THEN set_function
(DMM1, VOLT_AC)
    ELSE  IF ausdruck='GLEICHSTROMMESSUNG      ' THEN set_function
(DMM1, CURRENT_DC)
    ELSE  IF ausdruck='WECHSELSTROMMESSUNG     ' THEN set_function
(DMM1, CURRENT_AC)
    ELSE  IF ausdruck='WIDERSTANDSMESSUNG      ' THEN set_function
(DMM1, RESISTANCE_2W);
  set_range (DMM1, AUTOMATIC);
         { automatische Bereichswahl des Multimeters }
  set_settling_time (DMM1, YES);
         { Wartezeit zur Meßwertstabilisierung       }
  set_trig_source(DMM1, BUS);
         { Triggerung über IEC-Bus-Karte             }
  FOR i:=1 TO 32000 DO ;
  measure (DMM1, messwert);        { Ermittlung des Meßwerts   }
  ergebnis:=messwert;
  Messwertschreiben(ergebnis);
           { Schreiben des Meßwerts in Übergabedatei }

END.  { von example.pas }
```

7.4.5. helpme.pas

```
UNIT helpme;
  { enthält die Texte für die Online-Hilfe }

INTERFACE

PROCEDURE Hilfe;                { Menüprogramm }
PROCEDURE Menue_Hilfe1;         { Laden }
PROCEDURE Menue_Hilfe2;         { Erstellen }
PROCEDURE Menue_Hilfe3;         { Schreibe nach }
PROCEDURE Menue_Hilfe4;         { Drucken }
PROCEDURE Menue_Hilfe5;         { Start }
PROCEDURE Menue_Hilfe6;         { Zu MS-DOS }
PROCEDURE Menue_Hilfe7;         { Einzelmessung }
PROCEDURE Menue_Hilfe8;         { Meßreihe }
PROCEDURE Menue_Hilfe9;         { Selbsttest }
PROCEDURE Menue_Hilfe10;        { Dauertest }
PROCEDURE Menue_Hilfe11;        { Einzeltest }
PROCEDURE Menue_Hilfe12;        { Spannungen }


IMPLEMENTATION
USES Crt,Dos,global;
CONST screenimagesize=2000;
      maxwindow=5;
{$I fenster.pas}



PROCEDURE Hilfe;
VAR keychar:CHAR;
  BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' schon wieder vergessen ? ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Sie befinden sich jetzt im übergeordneten Bedienprogramm
des');
    WriteLn(' Baugruppenprüfgeräts WÖTRA B.');
    WriteLn;
    WriteLn(' Sie haben folgende Möglichkeiten, um einen Menüpunkt
auszuwählen :');
    WriteLn('');
    WriteLn('   - Drücken Sie den Anfangsbuchstaben des gewünschten
Menüpunkts');
    WriteLn('   - Bewegen Sie den Mauscursor auf einen Menüpunkt und
drücken');
    WriteLn('     Sie dann die linke Maustaste
');
    WriteLn('   - Drücken Sie F10
');
    WriteLn('
');
```

```
    WriteLn('  Befinden Sie sich dann in einem Pull-Down Menü, so
können Sie');
    WriteLn('  sich mit Hilfe der Maus oder der vier Cursortasten
einen ');
    WriteLn('  beliebigen Menüpunkt auswählen und diesen durch Drücken
von');
    WriteLn('  <RETURN> oder der linken Maustaste ablaufen lassen ');
    WriteLn;
    WriteLn('
---> <ESC> ');
    REPEAT
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
  END;  { von Hilfe }


{-------------------------------------------------------------------}

PROCEDURE Menue_Hilfe1;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' Prüfdatei Laden ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Mit diesem Menüpunkt haben Sie die Möglichkeit, eine
');
    WriteLn(' Prüfdatei zu laden. Wenn Sie den Punkt ausgewählt
haben,');
    WriteLn(' können Sie die gewünschte Datei entweder direkt
eingeben,');
    WriteLn(' oder durch drücken von <F3> eine komplette Übersicht
');
    WriteLn(' über die vorhandenen Prüfdateien erhalten.');
    WriteLn(' ');
    WriteLn(' Sie können sich mit Hilfe der vier Cursortasten in
dieser');
    WriteLn(' Übersicht bewegen.                              ');
    WriteLn;
    WriteLn(' Haben Sie die Markierung auf die gewünschte Datei
plaziert,');
    WriteLn(' so drücken Sie <RETURN> und die Datei wird übernommen
');
    WriteLn(' Anschließend wird die ausgewählte Datei angezeigt, und
Sie ');
    WriteLn(' können entweder bestätigen oder nochmal zurückspringen
');
    WriteLn(' und die Eingabe korrigieren');
    WriteLn;
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
```

```
    CloseWindow;
END;

{-----------------------------------------------------------------}

PROCEDURE Menue_Hilfe2;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' Prüfdatei Erstellen ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Mit diesem Menüpunkt haben Sie die Möglichkeit, eine
');
    WriteLn(' Prüfdatei zu erstellen oder zu ändern.');
    WriteLn(' ');
    WriteLn(' Es ist nicht notwendig, die zu ändernde Datei vorher ');
    WriteLn(' über den Menüpunkt <Prüfdatei Laden> einzulesen, da');
    WriteLn(' dies vom Unterprogramm <Erstellen> erledigt wird');
    WriteLn(' ');
    WriteLn(' Nach erfolgter Erstellung oder Änderung gelangen Sie');
    WriteLn(' automatisch wieder ins Hauptmenü!');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn;
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{-----------------------------------------------------------------}

PROCEDURE Menue_Hilfe3;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' Prüfdatei Schreiben nach ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Mit diesem Menüpunkt haben Sie die Möglichkeit, eine
');
    WriteLn(' Prüfdatei in ein anderes Laufwerk oder Verzeichnis zu
');
    WriteLn(' kopieren .');
    WriteLn(' ');
    WriteLn(' Dieser Programmpunkt benötigt für eine korrekte Arbeits-
');
```

```
    WriteLn(' weise folgende Angaben :');
    WriteLn(' ');
    WriteLn('  -Pfad und Dateiname der Quelldatei');
    WriteLn('  -Pfad und Dateiname der Zieldatei');
    WriteLn(' ');
    WriteLn(' Achtung !! ');
    WriteLn(' Vom Programm erfolgt keine Überprüfung, ob die Zieldatei
');
    WriteLn(' bereits existiert oder nicht. Gehen Sie also vorsichtig
vor,');
    WriteLn(' damit keine bestehenden Prüfdateien zerstört werden
!!');
    WriteLn;
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{-----------------------------------------------------------------}

PROCEDURE Menue_Hilfe4;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' Prüfdatei Drucken ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Mit diesem Menüpunkt haben Sie die Möglichkeit, eine
');
    WriteLn(' Prüfdatei auf dem angeschlossenen Drucker auszugeben.
');
    WriteLn(' ');
    WriteLn(' Die zu druckende Datei muß vorher mit dem
Unterprogramm');
    WriteLn(' <Laden> eingelesen werden.');
    WriteLn(' ');
    WriteLn(' Bevor Sie den Druckvorgang starten, sollten Sie sich');
    WriteLn(' vergewissern, daß der Drucker auch bereit ist, d.h.');
    WriteLn(' daß er angeschlossen, eingeschaltet und Papier
eingelegt');
    WriteLn(' ist.');
    WriteLn(' ');
    WriteLn(' Bitte beachten: ');
    WriteLn(' Solange der Ausdruck läuft, ist kein Arbeiten am WÖTRA
');
    WriteLn(' möglich ! Dies ist besonders bei sehr langen
Prüfdateien');
    WriteLn(' ausschlaggebend.');
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
```

```
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{-----------------------------------------------------------------}

PROCEDURE Menue_Hilfe5;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' Aktionen Start ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Dieser Menüpunkt muß aufgerufen werden, um eine Prüfung
');
    WriteLn(' zu starten. ');
    WriteLn(' ');
    WriteLn(' Die Prüfdatei, die abgearbeitet werden soll, muß vorher
');
    WriteLn(' mit dem Unterprogramm <Laden> eingelesen werden.');
    WriteLn(' ');
    WriteLn(' Nach dem Start läuft das Prüfprogramm selbständig ab;
ein');
    WriteLn(' Programmstop erfolgt nur dann, wenn entweder ein Fehler
an');
    WriteLn(' der Baugruppe auftritt, oder der Benutzer eine Warnung
er-');
    WriteLn(' halten soll.');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{-----------------------------------------------------------------}

PROCEDURE Menue_Hilfe6;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' Aktionen ZU MS-DOS ',5,12,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Bei Anwählung diese Menüpunkts wird dieses Programm
');
```

```
    WriteLn(' beendet und zum Betriebssystem MS-DOS zurückgekehrt. ');
    WriteLn;
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{------------------------------------------------------------------}

PROCEDURE Menue_Hilfe7;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' MESSUNG Einzelmessung ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Mit diesem Menüpunkt haben Sie die Möglichkeit, eine
');
    WriteLn(' Messung eines einzelnen Wertes mit Hilfe des
angeschlossenen ');
    WriteLn(' Digitalmultimeters durchzuführen.');
    WriteLn(' ');
    WriteLn(' Nach Anwählen dieses Punktes erscheint ein Menü,
durch');
    WriteLn(' das Sie die Art der Messung festlegen können. ');
    WriteLn(' Anschließend wird die Messung gestartet, und der
Meßwert');
    WriteLn(' direkt am Bildschirm ausgegeben');
    WriteLn(' ');
    WriteLn(' Bitte beachten:');
    WriteLn(' Da zu jeder Messung drei Festplattenzugriffe
erfolgen,');
    WriteLn(' dauert es ca. drei bis vier Sekunden, bevor der
Meßwert');
    WriteLn(' zur Verfügung steht.');
    WriteLn;
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{------------------------------------------------------------------}

PROCEDURE Menue_Hilfe8;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
```

```
    OpenWindow(' MESSUNG Meßreihe ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Mit diesem Menüpunkt haben Sie die Möglichkeit, eine
');
    WriteLn(' Meßreihe aufzustellen. ');
    WriteLn(' ');
    WriteLn(' Beim Aufruf erscheint zunächst ein Menü, durch welches
Sie');
    WriteLn(' die Art der Messung und die Anzahl der Meßwerte
festlegen können.');
    WriteLn(' ');
    WriteLn(' Bitte beachten:');
    WriteLn(' Da für jeden Meßwert drei Festplattenzugriffe erfolgen,
kann es');
    WriteLn(' erhebliche Zeit dauern, bis die komlette Meßreihe
vorliegt.');
    WriteLn(' Während dieser Zeit ist kein Arbeiten am WÖTRA möglich
!');
    WriteLn(' ');
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{-----------------------------------------------------------------}

PROCEDURE Menue_Hilfe9;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' TEST Selbsttest ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Führt den Selbsttest des WÖTRA durch.    ');
    WriteLn(' ');
    WriteLn(' Dieser Menüpunkt verlangt die Eingabe von Anfangs- und
');
    WriteLn(' Endknoten, für die der Selbsttest durchgeführt werden
soll.');
    WriteLn(' Es sind Knotennummern von 1-400 zulässig.');
    WriteLn(' ');
    WriteLn(' Bevor Sie den Selbsttest starten, müssen Sie
sicherstellen,');
    WriteLn(' daß die Kurzschlußstecker an den Ausgängen des WÖTRA
gesteckt');
    WriteLn(' sind.');
    WriteLn(' Tritt während des Selbsttests ein Fehler auf, so hält
das Programm');
    WriteLn(' an, und stellt Sie vor die Wahl, den Testschritt zu
wiederholen,');
```

```
    WriteLn(' den Fehler zu ignorieren, oder den gesamten Testlauf
abzubrechen.');
    WriteLn;
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{-------------------------------------------------------------}

PROCEDURE Menue_Hilfe10;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' TEST Dauertest ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Führt einen Dauertest des WÖTRA durch.    ');
    WriteLn(' ');
    WriteLn(' Dieser Menüpunkt verlangt die Eingabe von Anfangs-
und');
    WriteLn(' Endknoten, für die der Dauertest durchgeführt werden
soll.');
    WriteLn(' ');
    WriteLn(' Nach dem Starten des Tests werden alle Relais des
gewählten');
    WriteLn(' Bereichs nacheinander ein- und ausgeschaltet. Dies wird
solange ');
    WriteLn(' durchgeführt, bis vom Benutzer die Escape-Taste gedrückt
wird.');
    WriteLn(' Anschließend erscheinen in der Menüzeile die
Auswahlpunkte');
    WriteLn(' "Hauptmenü", "Neue Werte" zum Einlesen neuer
Testgrenzen,');
    WriteLn(' und "Weiter" zum Fortfahren des Tests an der Stelle, an
der ');
    WriteLn(' abgebrochen wurde.');
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{-------------------------------------------------------------}

PROCEDURE Menue_Hilfe11;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
```

```
    TextColor(black);
    OpenWindow(' TEST Einzeltest ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Mit diesem Menüpunkt haben Sie die Möglichkeit, bis
zu');
    WriteLn(' zehn Relais des WÖTRA einzeln anzusteuern. ');
    WriteLn(' ');
    WriteLn(' Das Programm verlangt die Eingabe von Knotennummer und
');
    WriteLn(' zugehöriger Relaisnummer. Es kann immer nur ein Relais
eines  ');
    WriteLn(' Knotens gesetzt werden.');
    WriteLn(' Nach jeder Relaiseingabe haben Sie die Möglichkeit,
entweder');
    WriteLn(' weitere Relais anzusteuern, oder ins Hauptmenü
zurückzukehren.');
    WriteLn(' Die Auswahl der gewünschten Option erfolgt gemäß der
Menüzeile.');
    WriteLn(' Falls Sie versuchen, ein zweites Relais eines Knotens zu
setzen, ');
    WriteLn(' so wird automatisch das zuerst gesetzte gelöscht. ');
    WriteLn(' ');
    WriteLn(' Wenn Sie zehn Relais gesetzt haben, werden Sie gefragt,
ob Sie');
    WriteLn(' ins Hauptmenü zurückkehren oder eine neue Folge von
Relais setzen');
    WriteLn(' wollen.');
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

{-----------------------------------------------------------------}

PROCEDURE Menue_Hilfe12;
VAR keychar:CHAR;
BEGIN
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow(' TEST Spannungen ',5,4,75,21);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn;
    WriteLn(' Dieser Menüpunkt überprüft, ob alle Versorgungs- und
Prüf-  ');
    WriteLn(' spannungen des WÖTRA korrekt anliegen.');
    WriteLn(' ');
    WriteLn(' Ergibt die Überprüfung der Spannungen keine Fehler, so
wird');
    WriteLn(' automatisch ins Hauptmenü zurückgesprungen. Wird ein
Fehler ');
```

```
    WriteLn(' festgestellt, so wird das Programm angehalten und der
Fehler');
    WriteLn(' ausgegeben. ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn(' ');
    WriteLn('
---> <ESC> ');
    REPEAT
      Cursor_Off;
      keychar:=ReadKey;
    UNTIL Ord(keychar)=27;
    CloseWindow;
END;

END.  { von helpme.pas }
```

7.4.6. pruefpr4.pas

```
PROGRAM Pruefprogramm_WoetraB ;
{$M $4000, 0, 10000 }

{ Übergeordnete Benutzeroberfläche des Wötra B. Dient zur Auswahl }
{ von Menüpunkten mit Hilfe von Maus und Tastatur.                }

USES Crt,Dos,Printer,          { Standardunits von Turbo-Pascal }
     Mausunit,                 { enthält die Mausfunktionen     }
     Objunit,                  { Proceduren zur Rahmenzeichnung }
     Selbst,                   { enthält die Selbsttestroutinen }
     Dauertes,                 { enthält die Dauertestroutinen  }
     Einzelte,                 { enthält die Einzeltestroutinen }
     Laden,                    { Proceduren zur Prüfdateiauswahl}
     Startpru,                 { Proceduren zum Prüfablauf      }
     Global,                   { Proceduren zur Relaissteuerung }
     Helpme ;                  { enthält die Hilfe-Prozeduren   }

CONST  screenimagesize=2000;              {Größe Hintergrundspeicher}
       maxwindow=5;                       {maximale Fensterzahl}

{$I fenster.pas}               { Include-File für Fenstertechnik }

TYPE menuearray=ARRAY[1..4,1..10] OF STRING[15] ;  { Menüpunkte  }
     menuearraymax=ARRAY[1..4] OF WORD;            { Menügröße   }


VAR dateiarray:menuearray;                  { Feld für Menüunterpunkte }
    dateiarraymax:menuearraymax;            { Feld für Menügröße       }
    auswahl:STRING[15];                     { ausgewählter Menüpunkt   }
    schalter:INTEGER;                       { Schalter für Menüauswahl }
    a,switch,switch1,wahl:INTEGER;

warnungsfenster,messfenster,fehlerfenster,kommentarfenster:fenster;
                                     {Fenster für Prüfablaufprogramm}
                                     {Definition in objunit.pas     }
{---------------------------------------------------------------------}

PROCEDURE Menue_Init ;
{ Initialisiert das Feld dateiarray, in dem die einzelnen }
{ Menüpunkte abgelegt sind                                }

  BEGIN
    dateiarraymax[1]:=4;         {Anzahl der Menüpunkte festlegen}
    dateiarraymax[2]:=2;
    dateiarraymax[3]:=2;
    dateiarraymax[4]:=4;
    dateiarray[1,1]:=' Laden          ';    {Menüpunkte festlegen}
    dateiarray[1,2]:=' Erstellen      ';
    dateiarray[1,3]:=' Schreibe nach  ';
    dateiarray[1,4]:=' Drucken        ';
    dateiarray[2,1]:=' Start        ';
    dateiarray[2,2]:=' Zu MS-DOS    ';
    dateiarray[2,3]:='              ';
    dateiarray[2,4]:='              ';
    dateiarray[3,1]:=' Einzelmessung  ';
```

```
    dateiarray[3,2]:=' Meßreihe        ';
    dateiarray[3,3]:='                 ';
    dateiarray[3,4]:='                 ';
    dateiarray[4,1]:=' Selbsttest      ';
    dateiarray[4,2]:=' Dauertest       ';
    dateiarray[4,3]:=' Einzeltest      ';
    dateiarray[4,4]:=' Spannungen      ';
  END;      { von Menue_Init }

{---------------------------------------------------------------------}

PROCEDURE Cur_Off ;
    { schaltet den Cursor aus, ohne dessen Position  }
    { zu verändern. }

VAR  register:REGISTERS ;
  BEGIN
     register.ah:=1;     { Funktionsnummer ins ah-Register }
     register.ch:=1;     { Cursor- Anfangszeile }
     register.cl:=0;     { Cursor-Endzeile      }
     Intr(16,register);  { Interrupt-Aufruf     }
  END;  { von Cursor_Off }

{---------------------------------------------------------------------}

PROCEDURE Cur_On;
   { schaltet den Cursor wieder an, ohne dessen Position }
   { zu verändern. }

VAR  register:REGISTERS ;
  BEGIN
    register.ah:=1;      { Funktionsnummer ins ah-Register }
    register.ch:=11;     { Cursor-Anfangszeile (Standardeinstellung) }
    register.cl:=12;     { Cursor-Endzeile (Standardeinstellung) }
    Intr(16,register);   { Interrupt-Aufruf }
  END;   { von Cur_On }

{---------------------------------------------------------------------}

PROCEDURE Oeffne_Fenster(k:INTEGER);
{ öffnet das Pull-Down Menü mit der Nummer k.  }

  BEGIN
  Verstecke_Maus;                        { Mauscursor unsichtbar }
    IF k=1 THEN                          { Fenster Datei }
      BEGIN
        TextBackground(lightgray);
        TextColor(black);
        OpenWindow('',1,2,16,5);         { Fenster öffnen }
        TextBackground(lightgray);       { Farben setzen }
        TextColor(black);
        WriteLn(dateiarray[1,1]);        { schreiben der Menüpunkte }
        WriteLn(dateiarray[1,2]);
        WriteLn(dateiarray[1,3]);
        Write(dateiarray[1,4]);
        GotoXY(1,1);TextBackground(black);TextColor(white);
        Write(dateiarray[1,1]);          { ersten Punkt hervorheben }
      END;
```

```
    IF k=2 THEN                          { Fenster Aktionen }
      BEGIN
        TextBackground(lightgray);
        TextColor(black);
        OpenWindow('',17,2,31,5);        { Fenster öffnen }
        TextBackground(lightgray);
        TextColor(black);
        WriteLn(dateiarray[2,1]);        { schreiben der Menüpunkte }
        WriteLn(dateiarray[2,2]);
        GotoXY(1,1);TextBackground(black);TextColor(white);
        Write(dateiarray[2,1]);          { ersten Punkt hervorheben }
      END;
    IF k=3 THEN                          { Fenster Messen }
      BEGIN
        TextBackground(lightgray);
        TextColor(black);                { Fenster öffnen }
        OpenWindow('',32,2,47,5);
        TextBackground(lightgray);
        TextColor(black);
        WriteLn(dateiarray[3,1]);        { Menüpunkte schreiben }
        WriteLn(dateiarray[3,2]);
        GotoXY(1,1);TextBackground(black);TextColor(white);
        Write(dateiarray[3,1]);          { ersten Punkt hervorheben }
      END;
    IF k=4 THEN                          { Fenster Test }
      BEGIN
        TextBackground(lightgray);
        TextColor(black);
        OpenWindow('',48,2,63,5);        { Fenster öffnen }
        TextBackground(lightgray);
        TextColor(black);
        WriteLn(dateiarray[4,1]);        { schreiben der Menüpunkte }
        WriteLn(dateiarray[4,2]);
        WriteLn(dateiarray[4,3]);
        Write(dateiarray[4,4]);
        GotoXY(1,1);TextBackground(black);TextColor(white);
        Write(dateiarray[4,1]);          { ersten Punkt hervorheben }
      END;
  Zeige_Maus;                            { Mauscursor sichtbar }
  END;          { von Oeffne_Fenster }


{-------------------------------------------------------------------}

PROCEDURE Menue_Hilfe;     { ruft je nach Bildschirmposition der }
                           { Maus die Hilfstexte auf             }
BEGIN
  IF Maus_Im_Bereich(16,16,128,17) THEN Menue_Hilfe1
    ELSE IF Maus_Im_Bereich(16,24,128,25) THEN Menue_Hilfe2
    ELSE IF Maus_Im_Bereich(16,32,128,33) THEN Menue_Hilfe3
    ELSE IF Maus_Im_Bereich(16,40,128,41) THEN Menue_Hilfe4
    ELSE IF Maus_Im_Bereich(144,16,248,17) THEN Menue_Hilfe5
    ELSE IF Maus_Im_Bereich(144,24,248,25) THEN Menue_Hilfe6
    ELSE IF Maus_Im_Bereich(264,16,384,17) THEN Menue_Hilfe7
    ELSE IF Maus_Im_Bereich(264,24,384,25) THEN Menue_Hilfe8
    ELSE IF Maus_Im_Bereich(392,16,512,17) THEN Menue_Hilfe9
    ELSE IF Maus_Im_Bereich(392,24,512,25) THEN Menue_Hilfe10
    ELSE IF Maus_Im_Bereich(392,32,512,33) THEN Menue_Hilfe11
```

```
    ELSE IF Maus_Im_Bereich(392,40,512,41) THEN Menue_Hilfe12
 END;  { von Menue_Hilfe }

{-------------------------------------------------------------------}

PROCEDURE Maus_Auswahl;
{ Auswahl eines beliebigen Menüpunktes mit Hilfe der Maus }
{ Übernahme des Menüpunktes mit RETURN oder linker        }
{ Maustaste.                                              }

VAR x1,y1,x2,y2:WORD;       { Koordinaten des Mauscursors }
    ymaus:WORD;
    statuslinks:BOOLEAN;    { Status der linken Maustaste }

  BEGIN
  Verstecke_Maus;
  REPEAT
    statuslinks:=FALSE;
    switch:=1;
    IF Maus_Im_Bereich(1,8,136,48) THEN
                            { Maus innerhalb des ersten }
      BEGIN                 { Menüfensters - Prüfdatei  }
        REPEAT
        Maus_Position(x2,y2);   { Mausposition ermitteln }
        switch1:=switch;        { alte Mausposition festhalten }
        IF Maus_Im_Bereich(16,8,112,9) THEN
                                {oberes Menüende erreicht }
          BEGIN
            switch:=4 ;
            GotoXY(1,4) ;             { Hardwarecursor versetzen }
            Setze_Maus(30,40) ;       { Mauscursor versetzen }
          END
        ELSE IF Maus_Im_Bereich(16,16,112,17) THEN
                                      { erster Menüpunkt }
          BEGIN                       { Prüfdatei laden   }
            switch:=1 ;
            GotoXY(1,1) ;             { Hardwarecursor nachführen }
          END
        ELSE IF Maus_Im_Bereich(16,24,112,25) THEN
                                      {zweiter Menüpunkt   }
          BEGIN                       {Prüfdatei erstellen }
            switch:=2 ;
            GotoXY(1,2) ;             { Hardwarecursor nachführen }
          END
        ELSE IF Maus_Im_Bereich(16,32,112,33) THEN
                                      { dritter Menüpunkt }
          BEGIN                       { Prüfdatei Schreibe nach }
            switch:=3 ;
            GotoXY(1,3) ;             { Hardwarecursor nachführen }
          END
        ELSE IF Maus_Im_Bereich(16,40,112,41) THEN
                                      { vierter Menüpunkt }
          BEGIN                       { Prüfdatei drucken }
            switch:=4 ;
            GotoXY(1,4) ;             { Hardwarecursor nachführen }
          END
        ELSE IF Maus_Im_Bereich(16,48,112,49) THEN
                                      { unteres Menüende }
```

```
        BEGIN
          switch:=1 ;
          GotoXY(1,1) ;                { Hardwarecursor nachführen }
          Setze_Maus(30,16) ;          { Mauscursor nachführen }
        END;
      statuslinks:=Knopf_Links;        { linke Maustaste abfragen }
      IF switch<>switch1 THEN          { alte und neue Mausposition }
        BEGIN                          { verschieden ? }
          TextBackground(lightgray);
          TextColor(black);            { alten Menübalken löschen }
          GotoXY(1,switch1);
          Write(dateiarray[a,switch1]);

          TextBackground(black);       { neuen Menübalken setzen }
          TextColor(white);
          GotoXY(1,switch);
          Write(dateiarray[a,switch]);
        END;
      UNTIL (x2<1) OR (x2>136) OR statuslinks OR KeyPressed ;
           { Schleife wird solange wiederholt, bis linkes oder }
           { rechtes Menüende erreicht ist oder eine Taste     }
           { oder die linke Maustaste gedrückt wird            }
      IF x2<1 THEN  a:=4 ;           { linkes Menüende  }
      IF x2>136 THEN a:=a+1 ;        { rechtes Menüende }
    END
  ELSE IF Maus_Im_Bereich(128,8,256,48) THEN
          { Maus innerhalb des zweiten Menüfensters; Ablauf sonst }
          { wie unter erstem Menüfenster beschrieben!             }
    BEGIN
      REPEAT
      Maus_Position(x2,y2);
      switch1:=switch;
      IF Maus_Im_Bereich(144,8,232,9) THEN
                                     { oberes Menüende }
        BEGIN
          switch:=2 ;
          GotoXY(1,2) ;
          Setze_Maus(200,24) ;
        END
      ELSE IF Maus_Im_Bereich(144,16,232,17) THEN
                                     { Menüpunkt Aktionen Start }
          BEGIN
            switch:=1 ;
            GotoXY(1,1) ;
          END
      ELSE IF Maus_Im_Bereich(144,24,232,25) THEN
                                     { Menüpunkt MS-DOS }
          BEGIN
            switch:=2 ;
            GotoXY(1,2) ;
          END
      ELSE IF Maus_Im_Bereich(144,32,232,33) THEN
                                     { unteres Menüende }
          BEGIN
            switch:=1 ;
            GotoXY(1,1) ;
            Setze_Maus(200,16) ;
          END;
```

```
        statuslinks:=Knopf_Links;
        IF switch<>switch1 THEN
          BEGIN
            TextBackground(lightgray);
            TextColor(black);
            GotoXY(1,switch1);
            Write(dateiarray[a,switch1]);

            TextBackground(black);
            TextColor(white);
            GotoXY(1,switch);
            Write(dateiarray[a,switch]);
          END;
        UNTIL (x2<128) OR (x2>256) OR statuslinks OR KeyPressed ;
        IF x2<128 THEN a:=a-1 ;
        IF x2>256 THEN a:=a+1 ;
      END
    ELSE IF Maus_Im_Bereich(248,8,384,48) THEN
          { Maus innerhalb des dritten Menüfensters; Ablauf sonst }
          { wie unter erstem Menüfenster beschrieben              }
      BEGIN
        REPEAT
        Maus_Position(x2,y2);
        switch1:=switch;
        IF Maus_Im_Bereich(264,8,368,9) THEN
                                      { oberes Menüende }
          BEGIN
            switch:=2 ;
            GotoXY(1,2) ;
            Setze_Maus(330,24) ;
          END
        ELSE IF Maus_Im_Bereich(264,16,368,17) THEN
                                       { Menüpunkt Einzelmessung }
          BEGIN
            switch:=1 ;
            GotoXY(1,1) ;
          END
        ELSE IF Maus_Im_Bereich(264,24,368,25) THEN
                                       { Menüpunkt Meßreihe }
          BEGIN
            switch:=2 ;
            GotoXY(1,2) ;
          END
        ELSE IF Maus_Im_Bereich(264,32,368,33) THEN
                                       { unteres Menüende }
          BEGIN
            switch:=1 ;
            GotoXY(1,1) ;
            Setze_Maus(330,16) ;
          END;
        statuslinks:=Knopf_Links;
        IF switch<>switch1 THEN
          BEGIN
            TextBackground(lightgray);
            TextColor(black);
            GotoXY(1,switch1);
            Write(dateiarray[a,switch1]);
```

```
        TextBackground(black);
        TextColor(white);
        GotoXY(1,switch);
        Write(dateiarray[a,switch]);
      END;
    UNTIL (x2<248) OR (x2>384) OR statuslinks OR KeyPressed ;
    IF x2<248 THEN a:=a-1 ;
    IF x2>384 THEN a:=a+1 ;
  END
ELSE IF Maus_Im_Bereich(376,16,512,48) THEN
      { Maus innerhalb des vierten Menüfensters; Ablauf sonst }
      { wie unter erstem Menüfenster beschrieben              }
  BEGIN
    REPEAT
    Maus_Position(x2,y2);
    switch1:=switch;
    IF Maus_Im_Bereich(392,8,496,9) THEN
                                 { oberes Menüende }
      BEGIN
        switch:=4 ;
        GotoXY(1,4) ;
        Setze_Maus(450,40) ;
      END
    ELSE IF Maus_Im_Bereich(392,16,496,17) THEN
                                 { Menüpunkt Selbsttest }
      BEGIN
        switch:=1 ;
        GotoXY(1,1) ;
      END
    ELSE IF Maus_Im_Bereich(392,24,496,25) THEN
                                 { Menüpunkt Dauertest }
      BEGIN
        switch:=2 ;
        GotoXY(1,2) ;
      END
    ELSE IF Maus_Im_Bereich(392,32,496,33) THEN
                                 { Menüpunkt Einzeltest }
      BEGIN
        switch:=3 ;
        GotoXY(1,3) ;
      END
    ELSE IF Maus_Im_Bereich(392,40,496,41) THEN
                                 { Menüpunkt Spannungen }
      BEGIN
        switch:=4 ;
        GotoXY(1,4) ;
      END
    ELSE IF Maus_Im_Bereich(392,48,496,49) THEN
                                 { unteres Menüende }
      BEGIN
        switch:=1 ;
        GotoXY(1,1) ;
        Setze_Maus(450,16) ;
      END;
    statuslinks:=Knopf_Links;
    IF switch<>switch1 THEN
      BEGIN
        TextBackground(lightgray);
```

```
            TextColor(black);
            GotoXY(1,switch1);
            Write(dateiarray[a,switch1]);

            TextBackground(black);
            TextColor(white);
            GotoXY(1,switch);
            Write(dateiarray[a,switch]);
          END;
          UNTIL (x2<376) OR (x2>512) OR statuslinks OR KeyPressed ;
        IF x2<376 THEN a:=a-1 ;
        IF x2>512 THEN a:=1 ;
      END
      ELSE statuslinks:=FALSE;

   IF ((NOT KeyPressed) AND (NOT statuslinks)) THEN
           { Ist keine Taste und die linke Maustaste nicht }
           { gedrückt, so hat der Mauscursor das aktuelle  }
           { Menüfenster verlassen; dieses muß geschlossen }
           { und das neue Menüfebster muß geöffnet werden  }
     BEGIN
       Verstecke_Maus;         { Mauscursor unsichtbar }
       CASE a OF               { Mauscursor ensprechend neuer }
         1:Setze_Maus(30,16); { Menuefensternummer setzen    }
         2:Setze_Maus(200,16);
         3:Setze_Maus(330,16);
         4:Setze_Maus(450,16);
       END;
       CloseWindow;            { altes Fenster schließen }
       Zeige_Maus;             { Mauscursor sichtbar     }
       Oeffne_Fenster(a);      { neues Fenster öffnen    }
     END;
   UNTIL statuslinks OR Keypressed;
                               { die Mausroutinen müssen solange }
                               { durchlaufen werden, bis entweder}
                               { eine Taste oder die linke Maus- }
                               { taste gedrückt wird             }
   IF NOT KeyPressed THEN  { es wurde die linke Maustaste betätigt }
     BEGIN
       auswahl:=dateiarray[a,WhereY]; { Menüpunkt übernehmen }
       schalter:=2;                   { Abbruchschalter setzen }
       CloseWindow;                   { aktuelles Fenster schließen }
     END;
   Zeige_Maus;
   END; { von Maus_Auswahl }

{-------------------------------------------------------------------}

PROCEDURE Cursor_Auswahl;
{ Diese Prozedur ermöglicht es, mit den vier Cursortasten  }
{ einen beliebigen Menüpunkt auszuwählen. Übername mit     }
{ <RETURN>                                                 }

VAR keychar :CHAR ;
    zeichen:INTEGER;
    x,y:WORD;

  BEGIN
```

```
    keychar:=ReadKey;
    IF ((Ord(keychar)=0) OR (Ord(keychar)=13)) THEN
                  { Prozedur nur dann ausführen, wenn entweder }
                  { RETURN oder oder eine Taste mit erweitertem}
                  { Tastaturcode betätigt wurde }
    BEGIN
    IF Ord(keychar)=13 THEN              { Return gedrückt }
      BEGIN
      auswahl:=dateiarray[a,WhereY];     { Übername des Menüpunkts }
      Verstecke_Maus;
      CloseWindow;                       { aktuelles Fenster schließen }
      Zeige_Maus;
      schalter:=2;                       { Schalter für Menüende setzen }
      Exit;                              { Sprung ins rufende Programm }
      END;
    keychar:=ReadKey;                    { Scancode lesen }
    zeichen:=Ord(keychar);
    Case zeichen OF                      { je nach Scancode Funktionen }
                                         { ausführen }
      72:BEGIN                           { Cursor auf }
         IF WhereY >= (hi(WindMin)) THEN
           BEGIN
           Verstecke_Maus;                   { Mauscursor unsichtbar }
           TextBackground(lightgray);
           TextColor(black);
           GotoXY(1,WhereY);
           Write(dateiarray[a,WhereY]);      { alten Menübalken löschen }
           GotoXY(1,WhereY-1);
           TextBackground(black);
           TextColor(white);
           Write(dateiarray[a,WhereY]);       { neuen Menübalken setzen }
           Maus_Position(x,y);                { Mausposition ermitteln }
           Setze_Maus(x,(WhereY+1)*8);        { Maus auf neuen Menüpunkt}
           switch:=WhereY;                    { platzieren              }
           Zeige_Maus;                        { Mauscursor sichtbar }
           END
         ELSE                                 { oberes Menüende }
           BEGIN
           Verstecke_Maus;                    { Beschreibung wie oben ! }
           TextBackground(lightgray);
           TextColor(black);
           GotoXY(1,WhereY);
           Write(dateiarray[a,WhereY]);
           GotoXY(1,dateiarraymax[a]);
           TextBackground(black);
           TextColor(white);
           Write(dateiarray[a,WhereY]);
           Maus_Position(x,y);
           Setze_Maus(x,(WhereY+1)*8);
           switch:=WhereY;
           Zeige_Maus;
           END
         END;
      80:BEGIN                                { Cursor ab }
         IF WhereY < dateiarraymax[a] THEN
           BEGIN                              { Beschreibung wie oben }
           Verstecke_Maus;
           TextBackground(lightgray);
```

```
          TextColor(black);
          GotoXY(1,WhereY);
          Write(dateiarray[a,WhereY]);
          GotoXY(1,WhereY+1);
          TextBackground(black);
          TextColor(white);
          Write(dateiarray[a,WhereY]);
          Maus_Position(x,y);
          Setze_Maus(x,(WhereY+1)*8);
          switch:=WhereY;
          Zeige_Maus;
          END
        ELSE                          { unteres Menüende }
          BEGIN
          Verstecke_Maus;             { Beschreibung wie oben }
          TextBackground(lightgray);
          TextColor(black);
          GotoXY(1,WhereY);
          Write(dateiarray[a,WhereY]);
          GotoXY(1,(hi(WindMin))-1);
          TextBackground(black);
          TextColor(white);
          Write(dateiarray[a,WhereY]);
          Maus_Position(x,y);
          Setze_Maus(x,(WhereY+1)*8);
          switch:=WhereY;
          Zeige_Maus;
          END
        END;
     75:BEGIN                        { Cursor links }
        a:=a-1;                      { neue Fensternummer }
        IF a<1 THEN a:=4;
        Verstecke_Maus;
        CloseWindow;                 { aktuelles Fenster schließen }
        Zeige_Maus;
        Oeffne_Fenster(a);           {neues Fenster öffnen }
        END;
     77:BEGIN                        { Cursor rechts }
        a:=a+1;                      { neue Fensternummer }
        IF a>4 THEN a:=1;
        Verstecke_Maus;
        CloseWindow;                 { aktuelles Fenster schließen }
        Zeige_Maus;
        Oeffne_Fenster(a);           { neues Fenster öffnen }
        END;
     59:BEGIN
        Menue_Hilfe;
        Verstecke_Maus;
        CloseWindow;
        Zeige_Maus;
        Oeffne_Fenster(a);
        END;
   END;
   Verstecke_Maus;
   CASE a OF                         { je nach geöffnetem Fenster}
    1:Setze_Maus(120,(WhereY+1)*8);  { Mauscursor neu setzen     }
    2:Setze_Maus(240,(WhereY+1)*8);
    3:Setze_Maus(376,(WhereY+1)*8);
```

```
     4:Setze_Maus(504,(WhereY+1)*8);
    END;
    Zeige_Maus;
  END;
  END;  { Cursor_Auswahl }


{--------------------------------------------------------------}

PROCEDURE Eingangsmenue ;
     { erzeugt Menü- und Statuszeile sowie }
     { das Arbeitsfenster                 }

VAR   keychar:CHAR ;

  BEGIN
    Window(1,1,80,1) ;
    TextBackground(lightgray);              { Menüzeile }
    ClrScr;
    TextColor(red);Write('   P');
    TextColor(black);Write('rüfdatei');
    TextColor(red);Write('       A');
    TextColor(black);Write('ktionen');
    TextColor(red);Write('       M');
    TextColor(black);Write('essen');
    TextColor(red);Write('       T');
    TextColor(black);Write('est');

    Window(1,25,80,25) ;                    { Statuszeile}
    TextBackground(lightgray);
    ClrScr;
    TextColor(red);Write('   F1');
    TextColor(black);Write('-Hilfe');
    TextColor(red);Write('     F10');
    TextColor(black);Write('-Menü');

    Window(1,2,80,24);                      { Arbeitsfenster }
    TextBackground(blue);
    ClrScr;
  END;   { von Eingangsmenue }

{--------------------------------------------------------------}

PROCEDURE Begruesung;
     { erzeugt zum Programmstart eine }
     { Begrüßungsmeldung              }

VAR keychar:CHAR;
  BEGIN
    TextColor(yellow);
    OpenWindow('',20,8,60,15);  { Begrüßungsfenster }
    TextBackground(blue);
    TextColor(yellow);
    WriteLn;
    WriteLn('  Prüfprogramm WÖTRA B  ');
    WriteLn;
    WriteLn('  Copyright 1989 Wolfgang Schraml');
    WriteLn;WriteLn;
```

```
    WriteLn('  Bitte Taste drücken');
    Cursor_Off;
    keychar:=ReadKey;
    CloseWindow;
    Window(1,2,80,24);
    Zeige_Maus;
  END;  { von Begruesung }

{-----------------------------------------------------------------}

PROCEDURE Menue_Auswahl ;
   { dient zur Auswahl eines Menüpunktes }
   { in der Menüzeile                    }

VAR keychar:CHAR ;
    x,y:WORD;              { Mauskoordinaten }
    zeichen:INTEGER;
    statuslinks:BOOLEAN;   { Status der linken Maustaste }
    unterschalter:BOOLEAN;

  BEGIN
  REPEAT
  unterschalter:=FALSE;
  Zeige_Maus;
    REPEAT
      statuslinks:=FALSE;
      Maus_Position(x,y);
      statuslinks:=Knopf_Links;
      IF statuslinks=TRUE THEN   { wenn linke Maustaste gedrückt }
        BEGIN
        IF Maus_Im_Bereich(24,0,96,1)=TRUE THEN zeichen:=80
                                   { Menü Prüfdatei }
          ELSE IF Maus_Im_Bereich(160,0,222,1)=TRUE THEN zeichen:=65
                                   { Menü Aktionen }
          ELSE IF Maus_Im_Bereich(286,0,330,1)=TRUE THEN zeichen:=77
                                   { Menü Messen }
          ELSE IF Maus_Im_Bereich(394,0,426,1)=TRUE THEN zeichen:=84
                                   { Menü Test }
          ELSE statuslinks:=FALSE ;
        END;
    UNTIL KeyPressed OR statuslinks ;  { bis Taste oder Maustaste }
    Delay(100);
    IF NOT statuslinks THEN           { wenn normale Taste }
      BEGIN
        keychar:=ReadKey ;
        IF Ord(keychar)=0 THEN keychar:=ReadKey;
        zeichen:=Ord(keychar);
      END;
    IF zeichen IN [84,116,65,97,80,112,77,109,59,68] THEN
unterschalter:=TRUE;
                                  { falls unerlaubte Eingabe }
    CASE zeichen OF               { Menüauswahl }
      84,116:BEGIN
         a:=4;
         Oeffne_Fenster(a);       { Menü Test <T> }
         END;
      65,97:BEGIN
         a:=2;
```

```
            Oeffne_Fenster(a);        { Menü Aktionen <A> }
            END;
         80,112:BEGIN
            a:=1;
            Oeffne_Fenster(a);        { Menü Prüfdatei <P> }
            END;
         77,109:BEGIN
            a:=3;
            Oeffne_Fenster(a);        { Menü Messen <M> }
            END;
         59:BEGIN                     { F1-Taste }
            Hilfe;
            unterschalter:=FALSE;
            END;
         68:BEGIN                     { F10-Taste }
            a:=1;
            Oeffne_Fenster(a);
            END;
         ELSE unterschalter:=FALSE;
       END;
     UNTIL unterschalter=TRUE;
     IF KeyPressed THEN keychar:=ReadKey;
     END; { Menü_Auswahl }

{-------------------------------------------------------------------}
{-------------------------------------------------------------------}

BEGIN          { HAUPTPROGRAMM }

Port_Programmieren;                { Programmierung des 8255 }
Relais_Reset;                      { Reset durchführen       }
Ausgeben($7fff,0);
Eingangsmenue;                     { Bildschirm aufbauen     }
Cur_Off;                           { Cursor abschalten       }
Begruesung;                        { Eingangsmeldung         }
Menue_Init;                        { Menüfeld aufbauen       }
REPEAT
  Reset_Maus;
  Aendere_Verhaeltnis(30,30);
  Zeige_Maus;
  schalter:=1;
  Eingangsmenue;
  Cur_Off;
  Menue_Auswahl;

  REPEAT                  { Schleife zur Menüpunktauswahl wird solange }
    Maus_Auswahl;         { durchlaufen, bis mit <RETURN> oder der     }
    IF KeyPressed THEN    { linken Maustaste eine Auswahl getroffen    }
      BEGIN               { wurde. }
        Cursor_Auswahl;
      END;
  UNTIL schalter=2;

  Verstecke_Maus;
  IF auswahl=' Laden          ' THEN wahl:=1
    ELSE IF auswahl=' Erstellen      ' THEN wahl:=2
    ELSE IF auswahl=' Schreibe nach ' THEN wahl:=3
    ELSE IF auswahl=' Drucken        ' THEN wahl:=4
```

```
    ELSE IF auswahl=' Start        ' THEN wahl:=5
    ELSE IF auswahl=' Zu MS-DOS    ' THEN wahl:=6
    ELSE IF auswahl=' Einzelmessung ' THEN wahl:=7
    ELSE IF auswahl=' Meßreihe      ' THEN wahl:=8
    ELSE IF auswahl=' Selbsttest    'THEN wahl:=9
    ELSE IF auswahl=' Dauertest     'THEN wahl:=10
    ELSE IF auswahl=' Einzeltest    'THEN wahl:=11
    ELSE IF auswahl=' Spannungen    'THEN wahl:=12;
  TextBackground(blue);
  TextColor(yellow);
  Window(1,2,80,24);
  CASE wahl OF
    1:BEGIN                    { Datei Laden }
      Cur_On;
      Lade;
      END;
    2:BEGIN                    { Datei erstellen }
      Cur_On;
      Exec('c:\tp5\diwö.exe','');
      END;
    3:BEGIN                    { hier muss die Prozedur für }
      END;                     { Datei Schreibe nach stehen }
    4:BEGIN                    { hier muss dei Prozedur für }
      END;                     { Datei drucken stehen       }
    5:BEGIN                    { Aktionen Start }
      Cur_On;
      Pruefung;                { Prüfung durchführen }
      END;
    6:BEGIN                    { Aktionen Zu MS-DOS }
      Cur_On;
      Verstecke_Maus;
      Window(1,1,80,25);
      TextBackground(white);
      TextColor(black);
      ClrScr;                  { Bildschirm in Grundzustand }
      EXIT;
      END;
    7:BEGIN              { hier muss die Prozedur für Einzelmessung }
      END;               { eingefügt werden. }
    8:BEGIN              { hier muss die Prozedur für Meßreihe    }
      END;               { eingefügt werden }
    9:BEGIN              { Test Selbsttest }
      Verstecke_Maus;
      Cur_On;
      Selbst_Test;
      Zeige_Maus;
      END;
   10:BEGIN              { Test Dauertest }
      Verstecke_Maus;
      Cur_On;
      Dauer_Test;
      Zeige_Maus;
      END;
   11:BEGIN              { Test Einzeltest }
      Verstecke_Maus;
      Cur_On;
      Einzel_Test;
      Zeige_Maus;
```

```
      END;
   12:BEGIN              { hier muß die Prozedur für Test Spannungen }
      END;               { eingefügt werden. }
  END;
UNTIL wahl=6;
Verstecke_Maus;

END.  { von pruefpr4.pas }
```

7.4.7. laden.pas

```
UNIT Laden ;
    { dient dem Laden von Prüfdateien. Wird von pruefpr4.pas }
    { verwendet. }

INTERFACE

PROCEDURE Lade;     { von pruefpr4.pas zugänglich }

IMPLEMENTATION

USES Crt,Dos,                       { Standardunits }
     filename,                      { enthält die Directory-Funktionen }
     global ;                       { allg. zugängliche Routinen }

CONST screenimagesize=2000;         { Voreinstellungen für fenster.pas }
      maxwindow=5;

{$I fenster.pas }

VAR    fname,actdir:string64;
       keychar:CHAR;
       schalter:BOOLEAN;

{------------------------------------------------------------------}

PROCEDURE Lade;

BEGIN
  Window(1,1,80,1);              { Menüzeile }
  TextBackground(lightgray);
  TextColor(black);
  ClrScr;
  Window(1,25,80,25);            { Statuszeile }
  ClrScr;
  Window(1,2,80,24);             { Arbeitsfenster }
  TextBackground(blue);TextColor(yellow);
  ClrScr;
  WriteLn('ëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë PRÜFDATEI LADEN
ëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë');

  REPEAT
    schalter:=FALSE;
    TextBackground(lightgray);
    TextColor(black);
    OpenWindow('',3,7,50,10);
    Textbackground(lightgray);
    TextColor(black);
    WriteLn('Geben Sie den Dateinamen und Pfad ein');
    WriteLn('oder drücken Sie <F3> !');
    REPEAT
      keychar:=ReadKey;
      IF Ord(keychar)=0 THEN
        BEGIN
          keychar:=ReadKey;
          IF Ord(keychar)=61 THEN       { F3 gedrückt }
```

```
            BEGIN
              CloseWindow;
              getdir(0,actdir);    { aktuelles Verzeichnis speichern }
              pruefname:=GetFileName('*.dat');
                                   { Verzeichnis einlesen }
              chdir(actdir);       { ins alte Verzeichnis zurück }
              schalter:=TRUE;
            END
          ELSE schalter:=FALSE;
        END
      ELSE
        BEGIN
          schalter:=TRUE;          { Datei über Tastatur eingeben }
          Write(keychar);
          ReadLn(pruefname);
          pruefname:=keychar+pruefname;
          CloseWindow;
        END;
    UNTIL schalter;

    TextBackground(lightgray);     { gewählte Datei anzeigen }
    TextColor(black);
    OpenWindow('',3,7,50,10);
    TextBackground(lightgray);
    TextColor(black);
    WriteLn(' Sie haben folgende Datei ausgewählt :');
    WriteLn(' ',pruefname);
    Write(' Einverstanden ? (j/n)');
    REPEAT
      keychar:=ReadKey;
    UNTIL (Ord(Upcase(keychar))=74) OR (Ord(Upcase(keychar))=78);
                               { warten, bis <J> oder <N> gedrückt }
    IF Ord(Upcase(keychar))=74 THEN schalter:=TRUE     { J gedrückt }
    ELSE schalter:=FALSE;                         { N gedrückt }
    CloseWindow;
  UNTIL schalter;
END;    { von Lade }

END.    { von Unit Laden }
```

7.4.8. startpru.pas

```
UNIT startpru;
{$M 4000, 0, 0}                { Einstellungen für Exec }
{$N+}                          { Koprozessor-Emulation  }

        { führt den Test der Baugruppe durch, deren Prüfdatei }
        { mit der Funktion Laden eingelesen wurde }

INTERFACE

PROCEDURE Pruefung;

IMPLEMENTATION

USES Crt,Dos,                   { Standardunits }
     Global,                    { allg. zugängliche Routinen }
     objunit;                   { Funktionen zur Rahmung }

CONST screenimagesize=2000;     { Voreinstellungen für fenster.pas }
      maxwindow=5;

{$I fenster.pas}                { fenster.pas einbinden }


TYPE     str23=STRING[23];

VAR datensatz:einzelschritt;    { in global definiert }
    i,koppelzaehler,k,spannungszaehler:SHORTINT;
    keychar:CHAR;
    ausgabename:STRING;
    messwert,messpannung:DOUBLE;
    fehlercode,code,umschalter,schritt:INTEGER;
    netzgeraet:REAL;
    pruefdatei:File OF einzelschritt;

{----------------------------------------------------------------------}

PROCEDURE Messartschreiben (name:STRING);
  { schreibt die gewünschte Meßart in die Datei paramet.dat, welche }
  { dann vom Meßprogramm example.exe ausgewertet wird }

  VAR f :TEXT;

  BEGIN
    Assign(f,'c:\mspascal\paramet.dat');      { Dateinamen zuordnen   }
    Rewrite(f);                                { zum Schreiben öffnen }
    Write(f,name);                             { Meßart schreiben     }
    Close(f);                                  { Datei schließen      }
  END;      {von Messartschreiben}

{----------------------------------------------------------------------}

PROCEDURE Messwertlesen (VAR messwertl:DOUBLE) ;
   { liest den vom Programm example.exe gemessenen und in die Datei }
   { messwert.dat geschriebenen Meßwert }
```

```
  VAR f:FILE OF DOUBLE ;

  BEGIN
    Assign(f,'c:\mspascal\messwert.dat');    { Dateinamen zuordnen  }
    Reset(f);                                { zum Lesen öffnen     }
    Read(f,messwert1);                       { Meßwert lesen        }
    Close(f);                                { Datei schließen      }
  END; {von Messwertlesen}

{----------------------------------------------------------------------}

PROCEDURE Knoten_Reset;
    { führt einen Reset für diejenigen Knoten durch, die während des }
    { aktuellen Prüfschritts gesetzt wurden }

BEGIN
  FOR i:=1 TO 8 DO
    BEGIN
      Ausgeben(datensatz.koppelpunkt[i]-1,0);
                                  { 8 Spannungsknoten löschen }
    END;
    Ausgeben(datensatz.messknoteneins-1,0);
                                  { Meßknoten 1 löschen }
    Ausgeben(datensatz.messknotenzwei-1,0);
                                  { Meßknoten 2 löschen }
    Umschaltung_Reset;            { Umschaltplatinen rücksetzen }
    Delay(50);                    { Verzögerung für Relais }
END;   { von Knoten_Reset }

{----------------------------------------------------------------------}

FUNCTION Testspannung:REAL;
   { Spannungswert vom A/D-Wandler übernehmen }

VAR i:WORD;
BEGIN
    testspannung:=SpannungLesen(9);
    testspannung:=((SpannungLesen(9)-2048)/204.8);
         { A/D-Wandler-Wert in tatsächlichen Spannungswert umrechnen }
END;   { von Testspannung }

{----------------------------------------------------------------------}

PROCEDURE Eingangs_Maske;
   { erstellt die Bildschirmmaske, die während der Prüfung}
   { vorhanden ist }

VAR warnungsfenster,kommentarfenster,messfenster:fenster;
           { Datentyp fenster in objunit.pas definiert }
  BEGIN
    TextBackground(lightgray);
    TextColor(black);
    Window(1,1,80,1);                  { Menüzeile }
    ClrScr;
    Window(1,25,80,25);                { Statuszeile }
    ClrScr;
    Window(1,2,80,24);                 { Arbeitsfenster }
    TextBackground(blue);
```

```
    TextColor(yellow);
    ClrScr;
    WriteLn;
    WriteLn('                Prüfprogramm  Baugruppe SN ',ausgabename,'
Schritt:');
    WriteLn('                ==================================    ');
    warnungsfenster.Init(7,15,71,18);    { Fenster für Warnungen }
    warnungsfenster.Zeigen;
    GotoXY(9,15);Write(' WARNUNGEN ');
    kommentarfenster.Init(7,20,71,23);   { Fenster für Kommentare }
    kommentarfenster.Zeigen;
    GotoXY(9,20);Write(' KOMMENTAR ');
    messfenster.Init(40,5,71,13);        { Fenster für Meßdurchgang }
    messfenster.Zeigen;
    GotoXY(42,5);Write(' MESSUNG ');     {Bildschirm beschriften }
    GotoXY(42,7);Write(' Sollwert:');
    GotoXy(42,8);Write(' Istwert :');
    GotoXy(42,10);Write(' Toleranz:');
    GotoXy(45,11);Write(' oben: ');
    GotoXy(45,12);Write('unten: ');
  END;  { von Eingangs_Maske }

{-----------------------------------------------------------------}

PROCEDURE Warnung_Halt;
      { hält das Prüfprogramm an kritischen Stellen an, und fordert }
      { den Benutzer auf die Registrierung der Warnung zu bestätigen}

    BEGIN
      GotoXy(8,16);
      FOR i:=1 TO 63 DO
        Write(datensatz.warnung[i]);
                         { erste Hälfte Warnung ausgeben }
      GotoXy(8,17);
      FOR i:=64 TO 126 DO
        Write(datensatz.warnung[i]);
                         { zweite Hälfte Warnung ausgeben }
      GotoXy(8,21);
      FOR i:=1 TO 63 DO
        Write(datensatz.kommentar[i]);
                         { erste Hälfte Kommentar ausgeben }
      GotoXy(8,22);
      FOR i:=64 TO 125 DO
        Write(datensatz.kommentar[i]);
                         { zweite Hälfte Kommentar ausgeben }
      TextBackground(red);       { neue Bildschirmfarben }
      TextColor(white);
      OpenWindow('',15,7,60,12); { Hinweisfenster öffnen }
      TextBackground(red);
      TextColor(white);
      ClrScr;                    { Fenster auf aktuelle Farbe setzen}
      WriteLn('Bitte beachten Sie die Warnung !!!');
      WriteLn;
      WriteLn('Wenn Sie bereit sind, mit der');
      WriteLn('Prüfung fortzufahren, so drücken Sie bitte');
      WriteLn('die <RETURN> Taste !');
      Cursor_Off;
      REPEAT
```

```
      keychar:=ReadKey;
    UNTIL Ord(keychar)=13;     { Programmhalt, bis Eingabe <RETURN> }
    CloseWindow;               { Hinweisfenster schließen }
    TextBackground(blue);
    TextColor(yellow);
    Window(1,2,80,24);         { alten Zustand wieder herstellen }
  END;          { von Warnung_Halt }

{-----------------------------------------------------------------}

PROCEDURE Fehler_Routine1;
   { frägt im Fehlerfall den Bediener, ob er abbrechen, wiederholen }
   { oder ignorieren will. Fehlerbehandlung bei Messungen.          }

       BEGIN
         TextBackground(red);
         TextColor(white);
         OpenWindow('Fehler',3,6,36,12);    { Hinweisfenster öffnen }
         TextBackground(red);
         TextColor(white);
         WriteLn('Meßwert außerhalb der Toleranz!!');
         WriteLn;WriteLn;
         WriteLn('Abbruch      --->A');
         WriteLn('Wiederholen  --->W');
         WriteLn('Ignorieren   --->I');
               REPEAT
                 keychar:=ReadKey;               { Wahl einlesen }
               UNTIL Ord(Upcase(keychar)) in [65,73,87];
               CASE Ord(Upcase(keychar)) OF
               65:BEGIN                          { Abbruch }
                  fehlercode:=2;
                  Window(1,1,80,25);
                  TextBackground(black);
                  TextColor(white);
                  ClrScr;
                  Exit;
                  END;
               73:BEGIN                          { Ignorieren }
                  fehlercode:=0;
                  END;
               87:BEGIN                          { Wiederholen }
                  fehlercode:=1;
                  END;
               END;

         CloseWindow;                   { Hinweisfenster schließen }
         TextBackground(blue);
         TextColor(yellow);
         Window(1,2,80,24);
       END;       { von Fehler_Routine1 }

{-----------------------------------------------------------------}

PROCEDURE Fehler_Routine2;
   { frägt im Fehlerfall den Bediener, ob er abbrechen, wiederholen }
   { oder ignorieren will. Fehler bei Durchgangsprüfung.            }

       BEGIN
```

```
          TextBackground(red);
          TextColor(white);
          OpenWindow('Fehler',3,6,36,12);    { Hinweisfenster öffnen }
          TextBackground(red);
          TextColor(white);
          WriteLn('Kein Durchgang zw. Knoten
',datensatz.messknoteneins);
          WriteLn('und Knoten ',datensatz.messknotenzwei);
          WriteLn;
          WriteLn('Abbruch      --->A');
          WriteLn('Wiederholen  --->W');
          WriteLn('Ignorieren   --->I');
                REPEAT
                  keychar:=ReadKey;                { Wahl einlesen }
                UNTIL Ord(Upcase(keychar)) in [65,73,87];
                CASE Ord(Upcase(keychar)) OF
                65:BEGIN                           { Abbruch }
                   fehlercode:=2;
                   Window(1,1,80,25);
                   TextBackground(black);
                   TextColor(white);
                   ClrScr;
                   Exit;
                   END;
                73:BEGIN                           { Ignorieren }
                   fehlercode:=0;
                   END;
                87:BEGIN                           { Wiederholen }
                   fehlercode:=1;
                   END;
                END;

          CloseWindow;                   { Hinweisfenster schließen }
          TextBackground(blue);
          TextColor(yellow);
          Window(1,2,80,24);
        END;      { von Fehler_Routine2 }

{-----------------------------------------------------------------}

PROCEDURE Fehler_Routine3;
   { frägt im Fehlerfall den Bediener, ob er abbrechen, wiederholen }
   { oder ignorieren will.Fehler bei Nicht-Durchgangsprüfung.       }

        BEGIN
          TextBackground(red);
          TextColor(white);
          OpenWindow('Fehler',3,6,36,12);    { Hinweisfenster öffnen }
          TextBackground(red);
          TextColor(white);
          WriteLn(' Durchgang zwischen ',datensatz.messknoteneins,' und
',datensatz.messknotenzwei);
          WriteLn;WriteLn;
          WriteLn('Abbruch      --->A');
          WriteLn('Wiederholen  --->W');
          WriteLn('Ignorieren   --->I');
                REPEAT
                  keychar:=ReadKey;                { Wahl einlesen }
```

```
                UNTIL Ord(Upcase(keychar)) in [65,73,87];
                CASE Ord(Upcase(keychar)) OF
                65:BEGIN                                 { Abbruch }
                   fehlercode:=2;
                   Window(1,1,80,25);
                   TextBackground(black);
                   TextColor(white);
                   ClrScr;
                   Exit;
                   END;
                73:BEGIN                                 { Ignorieren }
                   fehlercode:=0;
                   END;
                87:BEGIN                                 { Wiederholen }
                   fehlercode:=1;
                   END;
                END;

          CloseWindow;                        { Hinweisfenster schließen }
          TextBackground(blue);
          TextColor(yellow);
          Window(1,2,80,24);
        END;      { von Fehler_Routine3 }

{-------------------------------------------------------------------}

PROCEDURE Messung_Durchfuehren;
   { führt eine Messung am Wötra durch. Ruft das }
   { Programm example.exe auf }

BEGIN
              Messartschreiben(datensatz.messart);
              GotoXy(42,5);Write('    ',datensatz.messart,' ');
                                   { Meßart ausgeben }
              fehlercode:=0;
              REPEAT
                Exec('c:\mspascal\example.exe','');
                                   { Meßprogramm ausführen }
                Messwertlesen(messwert);
                                   { Meßwert lesen }
                GotoXy(53,8);Write(messwert:8:4);
                                   { Meßwert ausgeben }
                IF ((messwert<(datensatz.sollwert-
datensatz.toleranzunten))
                      OR
(messwert>(datensatz.sollwert+datensatz.toleranzoben))) THEN
                   { Abfrage, ob Meßwert innerhalb der vorgegebenen }
                   { Toleranzen liegt }
                  BEGIN
                    Fehler_Routine1;     { Fehlerbehandlung }
                  END
                ELSE fehlercode:=0;
              UNTIL ((fehlercode =0) OR (fehlercode=2));
END;      { von Messung_Durchführen }


{-----------------------------------------------------------------}
```

```
PROCEDURE Pruefung;
    { Prozedur für die eigentliche Prüfung. }
    { Wird von pruefpr4.pas aufgerufen }
BEGIN
  Port_Programmieren;       { Ausgabeport programmieren }
  Adport;                   { Einleseport programmieren }
  ausgabename:='';
  FOR i:=8 TO 14 DO   { Sachnummer aus Dateinamen und Pfad ermitteln }
  ausgabename:=ausgabename+pruefname[i];
  Eingangs_Maske;                { Bildschirmmaske aufbauen }
  Relais_Reset;                  { Wötra zurücksetzen }
  Knoten_Reset;                  { Meß- und Prüfknoten zurücksetzen }
REPEAT
  Assign(pruefdatei,pruefname);     { Dateinamen zuordnen }
  Reset(pruefdatei);                { Prüfdatei zum Lesen öffnen }
  TextBackground(lightgray);
  TextColor(black);
  OpenWindow('',15,7,60,12);
  TextBackground(lightgray);
  TextColor(black);
  WriteLn(' Geben Sie ein, ab welchem Schritt die Prüfung');
  Write(' erfolgen soll: ');
  Knoten_Lesen(schritt);             { Prüfschrittnummer einlesen }
  CloseWindow;
  TextBackGround(blue);
  TextColor(yellow);
  Window(1,2,80,24);
  seek(pruefdatei,schritt-1);        { Dateizeiger setzen }
  WHILE NOT Eof(pruefdatei) DO       { bis Ende Prüfdatei }
  BEGIN
  GotoXy(67,2);Write(filepos(pruefdatei)+1);
                          { Prüfschrittnr. ausgeben }
  Read(pruefdatei,datensatz);              { Datensatz einlesen }


  IF datensatz.messknoteneins<999 THEN    {normaler Pruefschritt}
    BEGIN
      GotoXy(8,16);
      FOR i:=1 TO 63 DO
        Write(datensatz.warnung[i]);      { erste Hälfte Warnung }
      GotoXy(8,17);
      FOR i:=64 TO 125 DO
        Write(datensatz.warnung[i]);      { zweite Hälfte Warnung }
      GotoXy(8,21);
      FOR i:=1 TO 63 DO
        Write(datensatz.kommentar[i]);    { erste Hälfte Kommentar }
      GotoXy(8,22);
      FOR i:=64 TO 125 DO
        Write(datensatz.kommentar[i]);    { zweite Hälfte Kommentar }

      FOR i:=1 TO 4 DO
        BEGIN
          IF ((datensatz.spannung[i]='  -  ') OR
(datensatz.spannung[i]='Leist'))
                THEN spannungszaehler:=spannungszaehler+1;
        END;
                { Ermittlung der Anzahl Spannungen }
```

```
      spannungszaehler:=4-spannungszaehler;
      IF spannungszaehler>1 THEN Ausgeben(402,15);
                              { R-Bus umschalten }

      FOR i:=1 TO 4  DO      { Schalten der max. 4 Spannungen }
        BEGIN
              umschalter:=0;
              IF datensatz.spannung[i]='220V¼' THEN umschalter:=1
                ELSE IF datensatz.spannung[i]='110V¼' THEN
umschalter:=2
                ELSE IF datensatz.spannung[i]=' 24V¼' THEN
umschalter:=4
                ELSE IF datensatz.spannung[i]=' 24V=' THEN
umschalter:=8
                ELSE IF datensatz.spannung[i]='Leist' THEN
umschalter:=111
                ELSE IF
datensatz.spannung[i][Length(datensatz.spannung[1])]='g' THEN
umschalter:=222
                ELSE IF datensatz.spannung[i]='  -  ' THEN i:=4;
              CASE umschalter OF
                111:BEGIN
                    Ausgeben(407,1);   { Leistungspunkt schalten }
                    i:=4;
                    END;
                222:BEGIN

Delete(datensatz.spannung[1],Length(datensatz.spannung[1])-1,2);
                    VAL(datensatz.spannung[1],netzgeraet,code);
                    { An dieser Stelle muß die Prozedur zur}
                    { Netzgeräteeinstellung }
                    { eingefügt werden }
                    Ausgeben(404+i-1,16);   { Netzgerät anlegen }
                    END;
                  1:BEGIN                { 220 V anlegen }
                    Ausgeben(404+i-1,1);
                    END;
                  2:BEGIN
                    Ausgeben(404+i-1,2);    { 110V anlegen }
                    END;
                  4:BEGIN
                    Ausgeben(404+i-1,4);    { 24V¼ anlegen }
                    END;
                  8:BEGIN
                    Ausgeben(404+i-1,8);     { 24V= anlegen }
                    END;
              END; { von Case }
        END;

      FOR i:=1 TO 8 DO      { Schalten der max. 8 Koppelpunkte }
        BEGIN
          IF ((datensatz.koppelpunkt[i]>0) AND
(datensatz.koppelpunkt[i]<401)) THEN
            BEGIN
              CASE i OF
              1: Ausgeben(datensatz.koppelpunkt[i]-1,4);
              2: Ausgeben(datensatz.koppelpunkt[i]-1,8);
              3: Ausgeben(datensatz.koppelpunkt[i]-1,1);
```

```
            4: Ausgeben(datensatz.koppelpunkt[i]-1,2);
            5: Ausgeben(datensatz.koppelpunkt[i]-1,4);
            6: Ausgeben(datensatz.koppelpunkt[i]-1,8);
            END;
          END
        ELSE i:=8 ;
      END;


        IF datensatz.messart='DURCHGANGSPRÜFUNG        ' THEN
          BEGIN
            GotoXy(42,5);Write('   DURCHGANGSPRÜFUNG     ');
            Ausgeben(403,8);    { 24V Prüfspannung auf R-Bus D0,D1 }
            Ausgeben(401,2);    { Durchgangsprüfung einschalten    }
            Ausgeben(datensatz.messknoteneins-1,1);
                        { 1.Meßknoten auf R-Bus D0 }
            Ausgeben(datensatz.messknotenzwei-1,16);
                        { 2.Meßknoten auf Meßleitung}
            fehlercode:=0;
            Delay(50);          { Verzögerung für Relais }
              REPEAT
                messpannung:=testspannung;   { Spannung lesen }
                IF (messpannung<4) OR (messpannung>6) THEN
                { im Fehlerfall wird der folgende Anweisungsblock }
                { ausgeführt                                       }
                  BEGIN
                    Fehler_Routine2;
                  END
                ELSE fehlercode:=0;
              UNTIL ((fehlercode=0) OR (fehlercode=2));
              IF fehlercode=2 THEN
                BEGIN
                  Knoten_Reset;   { Programmausstieg }
                  Exit;
                END;
              GotoXy(42,5);Write('áááááááááááááááááááááááááá');
              Knoten_Reset;
          END

        ELSE IF datensatz.messart='NICHTDURCHGANGSPRÜFUNG ' THEN
          BEGIN
            GotoXy(42,5);Write(' NICHTDURCHGANGSPRÜFUNG ');
            Ausgeben(403,8);    { 24V Prüfspannung auf R-Bus D0,D1 }
            Ausgeben(401,2);    { Durchgangsprüfung einschalten    }
            Ausgeben(datensatz.messknoteneins-1,1);
                        { 1.Meßknoten auf R-Bus D0 }
            Ausgeben(datensatz.messknotenzwei-1,16);
                        { 2.Meßknoten auf Meßleitung}
            fehlercode:=0;
            Delay(50);
              REPEAT
                messpannung:=0;
                messpannung:=testspannung;   { Spannung lesen }
                IF ((messpannung<-1) OR (messpannung>1)) THEN
                 { im Fehlerfall wird der folgende Anweisungsblock }
                 { ausgeführt                                       }
                  BEGIN
                    Fehler_Routine3;
```

```
                END
              ELSE fehlercode:=0;
            UNTIL ((fehlercode=0) OR (fehlercode=2));
            IF fehlercode=2 THEN
              BEGIN
                Knoten_Reset;
                Exit;
              END;
            GotoXy(42,5);Write('áááááááááááááááááááááááááááá');
            Knoten_Reset;
        END


      ELSE IF ((datensatz.messart='GLEICHSPANNUNGSMESSUNG ') OR
               (datensatz.messart='WECHSELSPANNUNGSMESSUNG') OR
               (datensatz.messart='GLEICHSTROMMESSUNG     ') OR
               (datensatz.messart='WECHSELSTROMMESSUNG    ') OR
               (datensatz.messart='WIDERSTANDSMESSUNG     ')) THEN
        BEGIN
          GotoXY(53,7);Write(datensatz.sollwert:8:4);
                    { Ausgabe der Vorgabewerte }
          GotoXy(53,11);Write(datensatz.toleranzoben:8:4);
          GotoXy(53,12);Write(datensatz.toleranzunten:8:4);
          Ausgeben(401,9);            { Meßgerät auf R-Bus D0,D1 }
          Ausgeben(datensatz.messknoteneins-1,1);
                                      { Meßknoten schalten }
          Ausgeben(datensatz.messknotenzwei-1,2);
          Messung_Durchfuehren;       { Meßwert ermitteln }
          IF fehlercode=2 THEN
              BEGIN
                Knoten_Reset;
                Exit;
              END;
          GotoXY(53,7);Write('         ');
                         { alte Werte am Schirm löschen }
          GotoXy(53,11);Write('         ');
          GotoXy(53,12);Write('         ');
          GotoXy(53,8);Write('         ');
          GotoXy(42,5);Write('áááááááááááááááááááááááááááá');
          Knoten_Reset;
        END


  END


ELSE   {Bediener warnen und halten}
  BEGIN
    Warnung_Halt;
  END;

END;

Close(pruefdatei);                    { Prüfdatei schließen }
Knoten_Reset;
TextBackground(lightgray);
TextColor(black);
OpenWindow('',15,7,60,12);
```

```
  TextBackground(lightgray);
  TextColor(black);
  WriteLn;
  WriteLn('Die Prüfung war erfolgreich !');
  WriteLn;
  WriteLn('Neuer Prüfdurchgang ---> < N >');
  WriteLn('Hauptmenü           ---> < H >');
  REPEAT
    keychar:=ReadKey;
  UNTIL Ord(Upcase(keychar)) in [72,78] ;
  CloseWindow;
UNTIL (Ord(Upcase(keychar))=72) ;

Relais_Reset;
END;


END. { Unit startpru.pas }
```

7.4.9. selbst.pas

```
UNIT selbst ;
        { UnterMenü Selbstest von pruefpr3.pas .Enthält }
        { Selbsttestroutine                              }

INTERFACE
          PROCEDURE Selbst_Test;

IMPLEMENTATION

USES Crt,Dos,                    { Standardunits }
     objunit,                    { Prozeduren zur Einrahmung }
     global;                     { allg. zugängliche Routinen }

CONST screenimagesize=2000;         { Voreinstellungen für fenster.pas }
      maxwindow=5;

{$I fenster.pas}                    { fenster.pas einbinden }

VAR adresse,nummer:WORD ;
    daten:BYTE;
    keychar:CHAR;
    eingabefenster,ausgabefenster:fenster;
    schalter:BOOLEAN;
    zahl,fehlercode:INTEGER;
    vonknoten,bisknoten:INTEGER;

{---------------------------------------------------------------------}

PROCEDURE Selbsttestmaske ;
{ zeichnet die Bildschirmmaske für den Selbst-}
{ test. Liest Start- und Endwert für den Test }
{ ein.                                        }

VAR fehler:WORD;
  BEGIN
    TextBackground(lightgray);    { Menüzeile }
    TextColor(black);
    Window(1,1,80,1);
    ClrScr;
    Window(1,25,80,25);           { Statuszeile }
    ClrScr;
    TextBackground(blue);
    TextColor(yellow);
    Window(1,2,80,24);            { Arbeitsfenster }
    ClrScr;
    WriteLn('ëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë SELBSTTEST
ëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë');
    eingabefenster.Init(2,4,60,8);
    eingabefenster.Zeigen;
    GotoXY(3,5);Write(' Test soll erfolgen :');
    GotoXY(3,6);Write(' von Knoten : ');
      REPEAT
        {$I-}                          { Fehlermeldungen abschalten }
        Knoten_Lesen(vonknoten);       { ersten Knoten einlesen }
        {$I+}                          { Fehlermeldungen einschalten }
```

```
        fehler:=IOResult;               { auf Falscheingabe pruefen }
        IF ((vonknoten>400) OR (vonknoten<1) OR (fehler<>0)) THEN
          BEGIN                                    { falsche Eingabe }
            GotoXY(17,6);Write('                      ');
            GotoXY(17,6);
          END;
      UNTIL ((vonknoten>0) AND (vonknoten<=400)
                  AND (fehler=0));            { bis Eingabe korrekt }
    GotoXY(3,7);Write(' bis Knoten : ');
      REPEAT
        {$I-}                           { Fehlermeldungen abschalten }
        Knoten_Lesen(bisknoten);        { zweiten Knoten einlesen }
        {$I+}                           { Fehlermeldungen einschalten }
        fehler:=IOResult;               { auf Falscheingabe pruefen }
        IF ((bisknoten<vonknoten) OR (bisknoten>400)
                                  OR (fehler<>0)) THEN
                                        { falsche Eingabe }
          BEGIN
            GotoXY(17,7);Write('                      ');
            GotoXY(17,7);
          END;
      UNTIL ((bisknoten>=vonknoten) AND (bisknoten<=400) AND
(fehler=0));
                        { Schleife wiederholen, bis Eingabe korrekt }

    IF vonknoten mod 2 = 0 THEN vonknoten:=vonknoten-1 ;
                                         { Knoten werden immer }
    IF bisknoten mod 2 <> 0 THEN bisknoten:=bisknoten+1 ;
                                         { paarweise geprüft   }
    ausgabefenster.Init(2,11,60,15);
    ausgabefenster.Zeigen;
                                 { Rahmen für Ausgabefenster }
    GotoXY(3,12);Write(' Testablauf : ');
                                 { Ausgabefenster einrichten }
    GotoXY(3,13);Write(' Knoten1: ');
    GotoXY(19,13);Write(' Relaisnummer: ');
    GotoXY(3,14);Write(' Knoten2: ');
    GotoXY(19,14);Write(' Relaisnummer: ');
  END;  { von Selbsttestmaske }

{-----------------------------------------------------------------}

FUNCTION Testspannung:REAL;
   { Spannungswert vom A/D-Wandler übernehmen }

VAR i:WORD;
BEGIN
    testspannung:=SpannungLesen(9);
    testspannung:=((SpannungLesen(9)-2048)/204.8);
         { A/D-Wandler-Wert in tatsächlichen Spannungswert umrechnen }
END;   { von Testspannung }

{----------------------------------------------------------------}

PROCEDURE Test_Ablauf ;
 { führt den Selbsttest durch    }

VAR i:INTEGER;
```

```
    messpannung:REAL;
    zaehler:WORD;

  BEGIN
    FOR zaehler:=(vonknoten-1) TO (bisknoten-2) DO
      BEGIN
        IF zaehler mod 2 = 0 THEN  { Knoten werden paarweise geprüft }
          BEGIN
            messpannung:=0;         { Meßspannung zurücksetzen }
            adresse:=zaehler ;
            Ausgeben(adresse+1,16);  { Meßleitungsrelais schalten }
            GotoXY(13,14);Write(adresse+2);
                                      { aktuelle Werte ausgeben }
            GotoXY(34,14);Write('5');
            daten:=1;                   { Relais Nr. 1 }
            nummer:=daten;              { Zaehler für Relaisnummer- }
                                               { ausgabe }
            WHILE daten<=8 DO                  { bis Relais Nr.4 }
              BEGIN
                Ausgeben(adresse,daten);       { Relais einschalten }
                GotoXY(13,13);Write(adresse+1);
                GotoXY(34,13);Write(nummer);
                Delay(100);                    { Verzögerung }
                REPEAT
                  messpannung:=testspannung;   { Spannung lesen }
                  IF (messpannung<4) OR (messpannung>6) THEN
                   { im Fehlerfall wird der folgende Anweisungsblock }
                   { ausgeführt                                      }
                    BEGIN
                      TextBackground(lightgray);
                      TextColor(black);
                      OpenWindow('',30,5,65,15);
                                { Fehlerfenster öffnen }
                      TextBackground(lightgray);
                      TextColor(black);
                      WriteLn('Achtung!! Knoten ',adresse+1,' Relais
',nummer);

                      WriteLn('kein Kontakt bei ein ');
                      WriteLn;
                      WriteLn('Abbruch     ---> A ');  { Fehlermenü }
                      WriteLn('Ignorieren  ---> I ');
                      WriteLn('Wiederholen ---> W ');
                      REPEAT
                        keychar:=ReadKey;           { Wahl einlesen }
                      UNTIL Ord(Upcase(keychar)) in [65,73,87];
                      CASE Ord(Upcase(keychar)) OF
                      65:BEGIN                          { Abbruch }
                         CloseWindow;
                         fehlercode:=1;
                         Exit;
                         END;
                      73:BEGIN                          { Ignorieren }
                         CloseWindow;
                         fehlercode:=0;
                         TextBackground(blue);
                         TextColor(yellow);
                         END;
                      87:BEGIN                          { Wiederholen }
```

```
                    CloseWindow;
                    fehlercode:=1;
                    END;
                 END;
                 TextBackground(blue);
                 TextColor(yellow);
                 Window(1,2,80,24);
                              { Arbeitsfenster wiederherstellen }
               END;
           UNTIL ((messpannung>4) AND (messpannung<6)) OR
(fehlercode=0);
           { Schleife solange durchlaufen, bis Spannung korrekt }
           { oder Prüfschritt ignoriert wird }
           Ausgeben(adresse,0);
                          { aktuelles Relais ausschalten }
           Delay(100);  { Verzögerung }
           REPEAT
             messpannung:=testspannung;   { Spannung lesen }
             IF (messpannung<-1) OR (messpannung>1) THEN
                 { im Fehlerfall folgenden Block ausführen }
               BEGIN
                 TextBackground(lightgray);
                 TextColor(black);
                 OpenWindow('',30,5,65,15); { Fehlerfenster }
                 TextBackground(lightgray);
                 TextColor(black);
                 WriteLn('Achtung!! Knoten ',adresse+1,' Relais
',nummer);
                 WriteLn('klebt bei aus ');
                 WriteLn;
                 WriteLn('Abbruch     ---> A '); { Fehlermenü }
                 WriteLn('Ignorieren  ---> I ');
                 WriteLn('Wiederholen ---> W ');
                 REPEAT
                   keychar:=ReadKey;
                 UNTIL Ord(Upcase(keychar)) in [65,73,87];
                 CASE Ord(Upcase(keychar)) OF
                 65:BEGIN                { Abbruch }
                    CloseWindow;
                    fehlercode:=1;
                    Exit;
                    END;
                 73:BEGIN                { Ignorieren }
                    CloseWindow;
                    fehlercode:=0;
                    TextBackground(blue);
                    TextColor(yellow);
                    END;
                 87:BEGIN                { Wiederholen }
                    CloseWindow;
                    fehlercode:=1;
                    END;
                 END;
                 TextBackground(blue);
                 TextColor(yellow);
                 Window(1,2,80,24);
               END;
           UNTIL ((messpannung>-1) AND (messpannung<1))
```

```
                              OR (fehlercode=0);
                     { Schleife solange durchlaufen, bis entweder }
                     { richtige Spannung oder Ignorieren }
                daten:=daten*2;        { nächstes Relais }
                nummer:=nummer+1;      { Relaisnummer erhöhen }
              END;

            Ausgeben(adresse,16);
                           { Meßleitungsrelais vom zweiten Knoten }
            GotoXY(13,13);Write(adresse+1);
            GotoXY(34,13);Write('5');
            daten:=1;                  { Relais Nr.1 }
            nummer:=daten;             { Relaisnummer }
            WHILE daten<=8 DO          { bis Relais Nr.4 }
              BEGIN
                Ausgeben(adresse+1,daten);  { Relais schalten }
                GotoXY(13,14);Write(adresse+2);
                GotoXY(34,14);Write(nummer);
                Delay(100);
                REPEAT
                  messpannung:=testspannung;  { Spannung lesen }
                  IF (messpannung<4) OR (messpannung>6) THEN
                      { auf korrekten Spannungswert prüfen }
                    BEGIN
                      TextBackground(lightgray);
                      TextColor(black);
                      OpenWindow('',30,5,65,15);  { Fehlerfenster }
                      TextBackground(lightgray);
                      TextColor(black);
                      WriteLn('Achtung!! Knoten ',adresse+2,' Relais
',nummer);

                      WriteLn('kein Kontakt bei ein ');
                      WriteLn;
                      WriteLn('Abbruch     ---> A ');
                      WriteLn('Ignorieren  ---> I ');
                      WriteLn('Wiederholen ---> W ');
                      REPEAT
                        keychar:=ReadKey;
                      UNTIL Ord(Upcase(keychar)) in [65,73,87];
                      CASE Ord(Upcase(keychar)) OF
                      65:BEGIN                  { Abbruch }
                         CloseWindow;
                         fehlercode:=1;
                         Exit;
                         END;
                      73:BEGIN                  { Ignorieren }
                         CloseWindow;
                         fehlercode:=0;
                         TextBackground(blue);
                         TextColor(yellow);
                         END;
                      87:BEGIN                  { Wiederholen }
                         CloseWindow;
                         fehlercode:=1;
                         END;
                      END;
                      TextBackground(blue);
                      TextColor(yellow);
```

```
                    Window(1,2,80,24);
                  END;
              UNTIL ((messpannung>4) AND (messpannung<6)) OR
(fehlercode=0);
                    { Schleife solange durchlaufen, bis entweder }
                    { korrekte Spannung oder Ignorieren }
              Ausgeben(adresse+1,0);   { aktuelles Relais löschen }
              Delay(100);
              REPEAT
                messpannung:=testspannung;  { Spannung lesen }
                IF (messpannung<-1) OR (messpannung>1) THEN
                   { prüfen, ob kein Durchgang }
                  BEGIN
                    TextBackground(lightgray);
                    TextColor(black);
                    OpenWindow('',30,5,65,15);  { Fehlerfenster }
                    TextBackground(lightgray);
                    TextColor(black);
                    WriteLn('Achtung!! Knoten ',adresse+2,' Relais
',nummer);
                    WriteLn('klebt bei aus ');
                    WriteLn;
                    WriteLn('Abbruch     ---> A ');
                    WriteLn('Ignorieren  ---> I ');
                    WriteLn('Wiederholen ---> W ');
                    REPEAT
                      keychar:=ReadKey;
                    UNTIL Ord(Upcase(keychar)) in [65,73,87];
                    CASE Ord(Upcase(keychar)) OF
                    65:BEGIN             { Abbruch }
                       CloseWindow;
                       fehlercode:=1;
                       Exit;
                       END;
                    73:BEGIN             { Ignorieren }
                       CloseWindow;
                       fehlercode:=0;
                       TextBackground(blue);
                       TextColor(yellow);
                       END;
                    87:BEGIN             { Wiederholen }
                       CloseWindow;
                       fehlercode:=1;
                       END;
                    END;
                    TextBackground(blue);
                    TextColor(yellow);
                    Window(1,2,80,24);
                  END;
              UNTIL ((messpannung>-1) AND (messpannung<1)) OR
(fehlercode=0);

              daten:=daten*2;           { nächstes Relais }
              nummer:=nummer+1;         { nächste Relaisnummer }
            END;

          Ausgeben(adresse,0);          { Relais löschen }
          Ausgeben(adresse+1,0);
```

```
          END;
       END;
    END;  { von Test_Ablauf }

{-------------------------------------------------------------------}

PROCEDURE Selbst_Test;
    { von pruefpr4.pas aufgerufene Prozedur }

BEGIN
  fehlercode:=0;
  Port_Programmieren;       { Ausgabebaustein programmieren }
  Relais_Reset;             { Wötra zurücksetzen }
  Adport;                   { A/D-Wandler Eingang programmieren }
  Selbsttestmaske;          { Bildscgirmmaske aufbauen }
  Ausgeben(400,15);         { Selbstestplatine schalten }
  Ausgeben(401,2);          { Durchgangsprüfung schalten}

  Test_Ablauf;
  IF fehlercode=0 THEN
    BEGIN
      TextBackground(lightgray);
      TextColor(black);
      OpenWindow('',30,5,65,12);
      TextBackground(lightgray);
      TextColor(black);
      WriteLn('Der Selbsttest war erfolgreich !');
      WriteLn;
      WriteLn('Drücken Sie eine Taste, um ins ');
      WriteLn('Hauptmenü zu gelangen !');
      REPEAT UNTIL KeyPressed;
      keychar:=ReadKey;
      CloseWindow;
    END;
  Relais_Reset;
END;

END.    { von UNIT selbst }
```

7.4.10. dauertes.pas

```
UNIT dauertes ;
        { UnterMenü Test von pruefpr3.pas .Enthält }
        { Dauertestroutine.                        }

INTERFACE

PROCEDURE Dauer_Test ;

{---------------------------------------------------------------------}

IMPLEMENTATION

USES Crt,Dos,            { Standardunits }
     objunit,            { Rahmenzeichnung }
     global;             { allg. zugängliche Routinen }


VAR adresse,nummer:WORD ;
    vonknoten,bisknoten:INTEGER;
    daten:BYTE;
    keychar:CHAR;
    eingabefenster,ausgabefenster:fenster;
    schalter:BOOLEAN;
    zahl:INTEGER;

{---------------------------------------------------------------------}


PROCEDURE Dauertestmaske ;
{ zeichnet die Bildschirmmaske für den Dauertest }
{ Liest Start- und Endwert für den Test ein.     }

VAR fehler:WORD;
  BEGIN
    TextBackground(lightgray);     { Menüzeile }
    TextColor(black);
    Window(1,1,80,1);
    ClrScr;
    Window(1,25,80,25);            { Statuszeile }
    ClrScr;
    TextColor(red);Write(' <ESC> - ');
    TextColor(black);Write('Abbruch');
    TextBackground(blue);
    TextColor(yellow);
    Window(1,2,80,24);             { Arbeitsfenster }
    ClrScr;
    WriteLn('ëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë DAUERTEST
ëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë');
    eingabefenster.Init(2,4,60,8);
    eingabefenster.Zeigen;
    GotoXY(3,5);Write(' Test soll erfolgen :');
    GotoXY(3,6);Write(' von Knoten : ');
      REPEAT
        {$I-}                       { Fehlermeldungen ausschalten }
        Knoten_Lesen(vonknoten);          { Anfangsknoten lesen }
```

```
        {$I+}                         { Fehlermeldungen einschalten }
        fehler:=IOResult;             { Fehlernummer ermitteln }
        IF ((vonknoten>400) OR (vonknoten<1) OR (fehler<>0)) THEN
          BEGIN                                    { falsche Eingabe }
            GotoXY(17,6);Write('                      ');
            GotoXY(17,6);
          END;
      UNTIL ((vonknoten>0) AND (vonknoten<=400)
               AND (fehler=0));                { bis Eingabe korrekt }
    GotoXY(3,7);Write(' bis Knoten : ');
      REPEAT
        {$I-}                         { Fehlermeldungen ausschalten }
        Knoten_Lesen(bisknoten);            { Endknoten lesen }
        {$I+}                         { Fehlermeldungen einschalten }
        fehler:=IOResult;
        IF ((bisknoten<vonknoten) OR (bisknoten>400)
                                 OR (fehler<>0)) THEN
                                               { falsche Eingabe }
          BEGIN
            GotoXY(17,7);Write('                      ');
            GotoXY(17,7);
          END;
      UNTIL ((bisknoten>=vonknoten) AND (bisknoten<=400) AND
(fehler=0));                             { bis Eingabe korrekt }
    ausgabefenster.Init(2,11,60,15);
    ausgabefenster.Zeigen;
    GotoXY(3,12);Write(' Testablauf : ');
    GotoXY(3,13);Write(' Knotennummer: ');
    GotoXY(3,14);Write(' Relaisnummer: ');
  END;

{-------------------------------------------------------------------}

PROCEDURE Test_Ablauf ;
 { führt den Dauertest durch, bis <ESC> }
 { gedrückt wird                        }

  BEGIN
    schalter:=FALSE;
    REPEAT
    FOR adresse:=(vonknoten-1) TO (bisknoten-1) DO
      BEGIN
        daten:=1;                                  { Relais 1 }
        nummer:=daten;
        WHILE daten<=16 DO                         { bis Relais Nr.5 }
          BEGIN
            Ausgeben(adresse,daten);             { Mitprotokollieren }
            GotoXY(17,13);Write('   ');          { am Monitor        }
            GotoXY(17,13);Write(adresse+1);
            GotoXY(17,14);Write('   ');
            GotoXY(17,14);Write(nummer);
              IF KeyPressed THEN
                IF Ord(ReadKey)=27 THEN          { ESC gedrückt }
                  BEGIN
                    Window(1,25,80,25);          { neue Statuszeile }
                    TextBackground(lightgray);
                    ClrScr;
                    Window(1,1,80,1);              { neue Menüzeile }
```

```
                    ClrScr;
                    TextBackground(lightgray);
                    TextColor(red);Write(' H');
                    TextColor(black);Write('auptmenü');
                    TextColor(red);Write('     W');
                    TextColor(black);Write('eiter');
                    TextColor(red);Write('     N');
                    TextColor(black);Write('eue Werte');
                    Cursor_Off;
                    REPEAT
                    zahl:=Ord(Upcase(ReadKey));
                    CASE zahl OF
                      72:BEGIN                         { Hauptmenü }
                         schalter:=FALSE;
                         EXIT;
                         END;
                      87:BEGIN                         { weiter }
                         Window(1,1,80,1);
                         TextBackground(lightgray); { alte Menüzeile }
                         ClrScr;
                         Window(1,25,80,25);       { alte Statuszeile }
                         TextColor(red);Write(' <ESC> - ');
                         TextColor(black);Write('Abbruch');
                         Window(1,2,80,24);
                         TextBackground(blue);
                         TextColor(yellow);
                         END;
                      78:BEGIN                         { neue Werte }
                         schalter:=TRUE;
                         EXIT;
                         END;
                    END;
                  UNTIL (zahl=72) OR (zahl=87) OR (zahl=78);
                  END;
            Delay(300);                               { Verzögerung }
            daten:=daten*2;                           { nächstes Relais }
            nummer:=nummer+1;
          END;
        Ausgeben(adresse,0);                        { alle Relais eines }
      END;                                          { Knoten löschen }
    UNTIL schalter;
  END;

{-------------------------------------------------------------------}

PROCEDURE Dauer_Test ;
   {  führt den Dauertest durch. }

BEGIN
  Port_Programmieren;        { Ausgabebaustein programmieren }
  Adport;                    { A/D-Wandler Eingang programmieren }
  REPEAT
    Dauertestmaske;          { Bildschirmmaske aufbauen }
    Relais_Reset;            { Wötra rücksetzen }
    Test_Ablauf;             { Test durchführen }
  UNTIL schalter=FALSE;
END;
END.   { von dauertes.pas }
```

7.4.11. einzelte.pas

```
UNIT einzelte ;
       { UnterMenü Test von pruefpr3.pas .Enthält }
       { Einzeltestroutinen                       }


INTERFACE
PROCEDURE Einzel_Test;       { von pruefpr4.pas aufgerufene Prozedur }


IMPLEMENTATION

USES Crt,Dos,
     objunit,                { Rahmenzeichnung }
     Global;                 { globale Definitionen und Routinen }

CONST screenimagesize=2000; { Voreinstellungen für fenster.pas }
      maxwindow=5;

{$I fenster.pas}             { Einbinden von fenster.pas }

VAR adresse,knotennummer,zahl,relaisnummer:INTEGER ;
    daten,k:BYTE;
    keychar:CHAR;
    eingabefenster,ausgabefenster:fenster;
    schalter,schalter1:BOOLEAN;

{-------------------------------------------------------------------}

PROCEDURE Maske;
  { erstellt die Bildschirmmaske für den Einzeltest }

  BEGIN
    TextBackground(lightgray);
    TextColor(black);
    Window(1,1,80,1);                       { Menüzeile }
    ClrScr;
    Window(1,25,80,25);                     { Statuszeile }
    ClrScr;
    TextBackground(blue);
    TextColor(yellow);
    Window(1,2,80,24);                      { Arbeitsfenster }
    ClrScr;
    WriteLn('ëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë EINZELTEST
ëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë');
    eingabefenster.Init(2,4,60,8);
    eingabefenster.Zeigen;
    ausgabefenster.Init(2,11,60,15);
    ausgabefenster.Zeigen;
    GotoXY(3,12);Write(' Testablauf : ');
    GotoXY(3,13);Write(' Knotennummer: ');
    GotoXY(3,14);Write(' Relaisnummer: ');
  END;    { von Maske }

{-------------------------------------------------------------------}
```

```
PROCEDURE Eingabe ;
{ zeichnet die Eingabemaske für den Einzel- }
{ test. Liest Knoten- und Relaisnummer ein  }

VAR fehler:WORD;
  BEGIN
    GotoXY(3,5);Write(k:2,'. Relais soll gesetzt werden :');
    GotoXY(3,6);Write(' Knotennummer : ');
    GotoXY(19,6);Write('            ');GotoXY(19,6);
      REPEAT
        {$I-}                                { Fehlermeldungen aus }
        Knoten_Lesen(knotennummer);          { Wert lesen }
        {$I+}                                { Fehlermeldungen ein }
        fehler:=IOResult;
        IF ((knotennummer>400) OR (knotennummer<1) OR (fehler<>0))
THEN
                    { ungültige Eingabe }
          BEGIN
            GotoXY(19,6);Write('                          ');
            GotoXY(19,6);
          END;
      UNTIL ((knotennummer>0) AND (knotennummer<=400) AND (fehler=0));
                   { bis Eingabe korrekt }
    GotoXY(3,7);Write(' Relaisnummer : ');
    GotoXY(19,7);Write('          ');GotoXY(19,7);
      REPEAT
        {$I-}                                { Fehlermeldungen aus }
        Knoten_Lesen(relaisnummer);          { Wert lesen }
        {$I+}                                { Fehlermeldungen ein }
        fehler:=IOResult;
        IF ((relaisnummer<1) OR (relaisnummer>5) OR (fehler<>0)) THEN
                                          { ungültige Eingabe }
          BEGIN
            GotoXY(19,7);Write('                          ');
            GotoXY(19,7);
          END;
      UNTIL ((relaisnummer>0) AND (relaisnummer<=5) AND (fehler=0));
                                          { bis Eingabe korrekt }
  END;   { von Eingabe }

{-------------------------------------------------------------------}

PROCEDURE Ausgabe;
   { gibt die Nummer des gesetzten Relais am Bildschirm }
   { zur Kontrolle aus                                   }

  BEGIN
      GotoXY(17+k*4,13);Write(adresse+1);     { Ausgabe der Werte }
      GotoXY(17+k*4,14);Write(relaisnummer);
      TextBackground(lightgray);
      Window(1,1,80,1);                       { neue Menüzeile }
      ClrScr;
      TextBackground(lightgray);
      TextColor(red);Write(' H');
      TextColor(black);Write('auptmenü');
      TextColor(red);Write('    W');
      TextColor(black);Write('eiter');
```

```
      Cursor_Off;
   END;  { von Ausgabe }

{--------------------------------------------------------------------}

PROCEDURE Auswahl;
   { führt die Auswahl zwischen den Menüpunkten Weiter und }
   { Rückkehr ins hauptmenü durch. }

  BEGIN
    REPEAT
      zahl:=Ord(Upcase(ReadKey));
    UNTIL (zahl=72) OR (zahl=87);
    CASE zahl OF                              { Tastencode einlesen }
      72:BEGIN                                { Hauptmenü }
         schalter:=FALSE;
         EXIT;                            { ins übergeordnete Programm }
         END;
      87:BEGIN                                { weiter }
         Window(1,1,80,1);                    { alte Menüzeile }
         TextBackground(lightgray);
         ClrScr;
         Window(1,2,80,24);
         TextBackground(blue);
         TextColor(yellow);
         schalter:=TRUE;                      { kein Abbruch }
         END;
    END;
  END;      { von Auswahl }

{--------------------------------------------------------------------}

PROCEDURE Test_Ablauf ;
 { führt den Einzeltest durch, bis <ESC> }
 { gedrückt wird                         }

  BEGIN
      adresse:=knotennummer-1;                 { Adresse bestimmen }
      CASE relaisnummer OF                     { Daten bestimmen }
        1:daten:=1;
        2:daten:=2;
        3:daten:=4;
        4:daten:=8;
        5:daten:=16;
      END;
      Ausgeben(adresse,daten);                  { Ausgabe }
      Delay(300);
  END;  { von Test_Ablauf }

{--------------------------------------------------------------------}

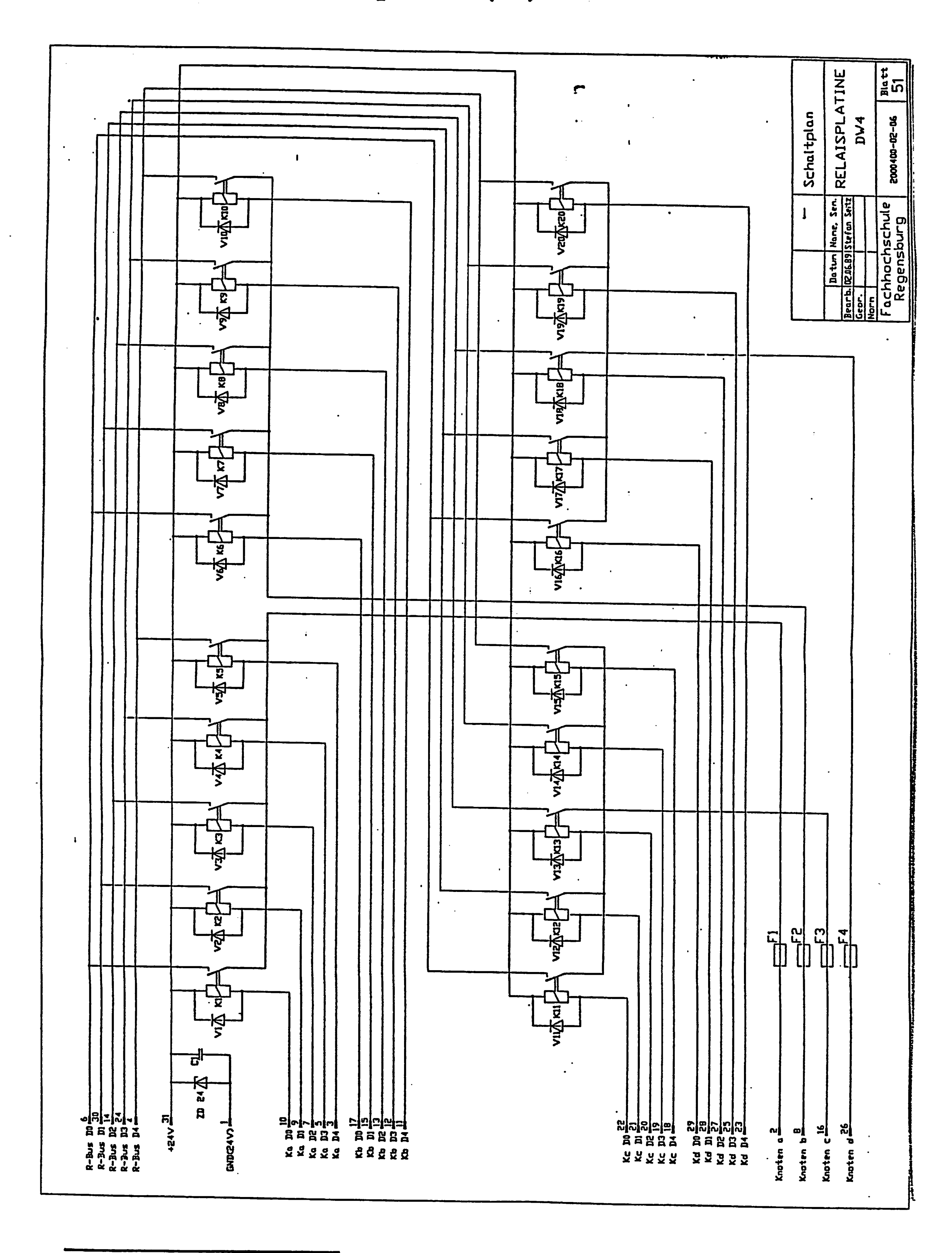
PROCEDURE Einzel_Test;
   { führt den Einzeltest durch. Wird von pruefpr4.pas aufgerufen }

BEGIN
  Port_Programmieren;                          { Port vorbereiten }
  Adport;
  REPEAT
```

```
    schalter1:=FALSE;
    Maske;                              { Bildschirmmaske }
    Relais_Reset;                       { alle Relais löschen }
    k:=0;
    schalter:=TRUE;
    REPEAT
      k:=k+1;                           { Relaiszähler erhöhen }
      Eingabe;                          { Parameter einlesen }
      Test_Ablauf;                      { Relais setzen }
      Ausgabe;                          { Relaiszustand ausgeben }
      Auswahl;
      IF k=10 THEN                      { Ende erreicht }
        BEGIN
          TextBackground(lightgray);
          TextColor(black);
          OpenWindow('',2,8,60,12);     { Warnung ausgeben }
          TextColor(black);
          TextBackground(lightgray);
          WriteLn('  Es dürfen nur maximal 10 Relais gesetzt werden!
');
          WriteLn; WriteLn;
          WriteLn('  weiter --> RETURN');
          Cursor_Off;
          REPEAT UNTIL KeyPressed;
          keychar:=ReadKey;
          CloseWindow;
          Window(1,1,80,1);             { neue Menüzeile }
          TextBackground(lightgray);
          TextColor(red);Write(' H');
          TextColor(black);Write('auptmenü');
          Textcolor(red);Write('     N');
          TextColor(black);Write('eu setzen');
          Cursor_Off;
          REPEAT
            zahl:= Ord(Upcase(ReadKey));
          UNTIL (zahl=72) OR (zahl=78);
          CASE zahl OF
            72:BEGIN                        { Hauptmenü }
               schalter:=FALSE;
               END;
            78:BEGIN                        { Einzeltest neu starten }
               schalter1:=TRUE;
               END;
          END;
        END;
    UNTIL (schalter=FALSE) OR (schalter1=TRUE);
                  { gewünschter Abbruch oder 10 Relais gesetzt }
    IF schalter=FALSE THEN                  { zum Hauptmenü }
      BEGIN
        Relais_Reset;
        Exit;                               { Abbruch }
      END;
  UNTIL schalter1=FALSE;        { Abfrage ob Einzeltest neu starten}
END;   { von Einzel_Test }

END.   { von Unit einzelte.pas }
```

7.5. Schaltplan Relaisplatine (6*)



(6*) siehe Diplomarbeit Stefan Seitz

7.6. Datenblatt U/I-Konverter 1B21



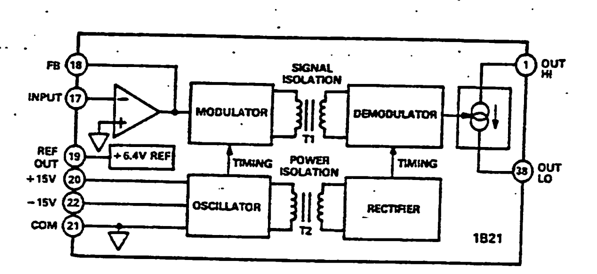
Isolated, Loop-Powered Voltage-to-Current Converter

1B21

FEATURES
- Wide Input Range: 0-1V to 0-10V
- High CMV Isolation: 1500V rms
- Programmable Output Ranges: 4mA to 20mA
  0 to 20mA
- Load Resistance Range: 0 to 1.35kΩ max
- High Accuracy
  - Low Offset Tempco: ±300nA/°C
  - Low Gain Tempco: ±50ppm/°C
  - Low Nonlinearity: ±0.02%
  - High CMR: 90dB min
- Small Package: 0.7"×2.1"×0.35"
- Meets IEEE Std. 472: Transient Protection (SWC)

APPLICATIONS
- Multichannel Process Control
- D/A Converter – Current Loop Interface
- Analog Transmitters and Controllers
- Remote Data Acquisition Systems

1B21 FUNCTIONAL BLOCK DIAGRAM



GENERAL DESCRIPTION

The 1B21 is an isolated voltage-to-current converter that incorporates a unique circuit design utilizing transformer based isolation and automated surface mount manufacturing technology. It provides an unbeatable combination of versatility and performance in a compact plastic package. Designed for industrial applications, it is especially suited for harsh environments with extremely high common-mode interference.

Functionally, the V/I converter consists of four basic sections: input conditioning, modulator, demodulator and current source (1B21 Functional Block Diagram). The input is a resistor programmable gain stage that accepts a 0-1V to 0-10V voltage input. This maps into a 0 to 20mA output or can be offset by 20% using the internal reference for 4mA to 20mA operation. The high level signal is modulated and passed across the barrier which provides complete input to output galvanic isolation of 1500V rms continuous by the use of transformer coupling techniques. Nonlinearity is an excellent ±0.05% max.

Designed for multichannel applications, the 1B21 requires an external loop supply and can accept up to 30V max. This would provide a loop compliance of 27V, which is sufficient to drive a 1.35kΩ load resistance.

The 1B21 is fully specified over −25°C to +85°C and operates over the industrial (−40°C to +85°C) temperature range.

DESIGN FEATURES AND USER BENEFITS

High CMV Isolation: The 1B21 features high input to output galvanic isolation to eliminate ground loops and offer protection against damage from transients and fault voltages. The isolation barrier will withstand continuous CMV of 1500V rms and meets the IEEE Standard for Transient Voltage Protection (Std. 472-SWC).

Small Size: The 1B21 package size (0.7"×2.1" DIP) makes it an excellent choice in multichannel systems for maximum channel density. The 0.35" height also facilitates applications with limited board clearance.

Ease of Use: Complete isolated voltage-to-current conversion with minimum external parts required to get a conditioned current signal. No external buffers or drivers are required.

11

SPECIFICATIONS (typical at +25°C and V_S = ±15V unless otherwise noted)

Model	1B21AN
INPUT SPECIFICATIONS	
Input Range	0 to +10V
Full-Scale Input	+1V min to +10V max
Input Bias Current	±30pA (±400pA max)
OUTPUT SPECIFICATIONS	
Current Output Range	4mA to 20mA, 0 to 20mA
Load Compliance at V_{LOOP} = 30V	27V min
Max Output Current @ Input Overload	25mA
Output Noise, 100Hz Bandwidth	1µA p-p
NONLINEARITY (% of Span)	±0.02% (±0.05% max)
ISOLATION	
CMV, Input to Output Continuous	1500V rms
CMR, @ 60Hz	90dB min
Transient Protection	IEEE-STD 472 (SWC)
ACCURACY	
Warm-Up Time to Rated Performance	5 min
Total Output Error @ +25°C (Untrimmed)	
Offset (V_{IN} = 0V)[1]	±100µA
Span (V_{IN} = +10V)	±0.6% FSR
vs. Temperature (−25°C to +85°C)	
Offset[2]	±300nA/°C
Span	±50ppm/°C
REFERENCE OUTPUT	
Voltage	+6.4V dc
Output Error	±1.5% max
Temperature Coefficient	±20ppm/°C max
DYNAMIC RESPONSE	
Settling Time to 0.1% of F.S. for 10V Step	9ms
Small Signal Bandwidth	100Hz
POWER SUPPLY	
Input Side	
Operating Voltage	±15V ±5%
Quiescent Current	
+15V Supply	10mA
−15V Supply	5mA
Power Supply Rejection	±0.01%/V
Loop Side	
Operating Voltage	+15V to +30V
Maximum Current	25mA
ENVIRONMENTAL	
Temperature Range	
Rated Performance	−25°C to +85°C
Operating	−40°C to +85°C
Storage	−40°C to +85°C
Relative Humidity, Noncondensing	0 to 95% @ +60°C
CASE SIZE	0.7" × 2.1" × 0.35" (17.8 × 53.3 × 8.9)mm

NOTES
[1]For 0-20mA mode. For 4-20mA mode an additional 60µA is contributed by the ±1.5% reference error on the 4mA output.
2For a complete discussion of the temperature effects of the offset resistor and reference refer to "Using the 1B21" section.
Specifications subject to change without notice.

OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

AC1060 MATING SOCKET

PIN DESIGNATIONS

PIN	FUNCTION
1	OUT HI
17	IN
18	FB
19	REF
20	+15V
21	COM
22	−15V
38	OUT LOW

INSIDE THE 1B21

Referring to the functional block diagram, the ±15V power inputs provide power to both the input side circuitry and the power oscillator. The 25kHz power oscillator provides both the timing information for the signal modulator and drives transformer T2 for the output side power supplies. The secondary winding of T2 is full wave rectified and filtered to create the output side power.

The input stage is configured as an inverting amplifier with three user supplied resistors for-gain, offset and feedback. The conditioned signal is modulated to generate a square wave with a peak-to-peak amplitude proportional to V_{IN}. This signal drives the signal transformer T1. An internal reference with a nominal output voltage of +6.4V and tempco of ±20ppm/°C is provided to develop a 4mA offset for 4mA to 20mA current loop applications.

After passing through signal transformer T1, the amplitude modulated signal is demodulated and filtered by a single pole filter. Timing information for the output side is derived from the power transformer T2. The filtered output provides the control signal for the voltage-to-current converter stage. An external power supply is required in series with the load to complete the current loop.

USING THE 1B21

Input Configurations: The 1B21 has been designed with a flexible input stage for a variety of input and output ranges. The basic interconnection for setting gain and offset is shown in Figure 1. The output of the internal amplifier is constrained to 0 to −5V, which maps into 0 to 20mA across the isolation barrier. Thus to create a 4mA offset at the output, the input amplifier has to be offset by 1V.

For example, for 0 to 20mA operation the transfer function for the input stage is:

$$5/V_{IN} = R_F/R_I$$

and no offset resistor is needed. For 4mA to 20mA operation we get:

$$4/V_{IN} = R_F/R_I$$

which maps the input voltage into a 4V span. To create a 1V offset at the output of the internal amplifier (4mA at the output of the 1B21) a current derived from the reference can be fed into the summing node. The offset resistor (for a 1V output offset) will be given by the equation: $R_O = 6.4R_F$. For most applications it is recommended that R_F be in the 25kΩ ±20% range. Resistor values for typical input and output ranges are shown in Table I.

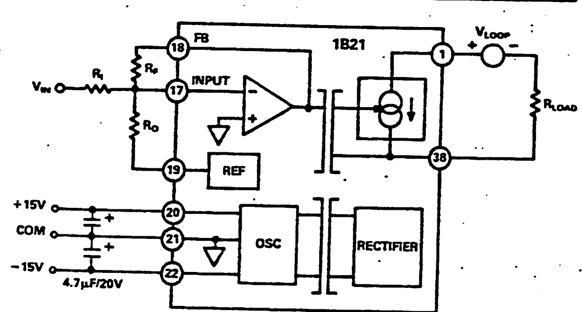


Figure 1. Basic Interconnections

Input Volts	Output mA	R_I kΩ	R_F kΩ	R_O kΩ
0-5	0-20	25	25	Open
0-10	0-20	50	25	Open
0-5	4-20	25	20	128
0-10	4-20	50	20	128
1-5	4-20	25	25	Open

Table I. Resistor Values for Typical Ranges

Adjustments: Figure 2 is an example of using potentiometers for trimming gain and offset for a 0-5V input and 0 to 20mA output. The network for offset adjustment keeps the resistors relatively small to minimize noise effects while giving a sensitivity of ±1% of span. For more adjustment range, resistors smaller than 274k can be used. Resistor values from Table I can be substituted for other input and output ranges.

In general, any bipolar voltage can be input to the 1B21 as long as it is offset to meet the 0 to −5V constraint of the modulator and the input signal range is 1V minimum.

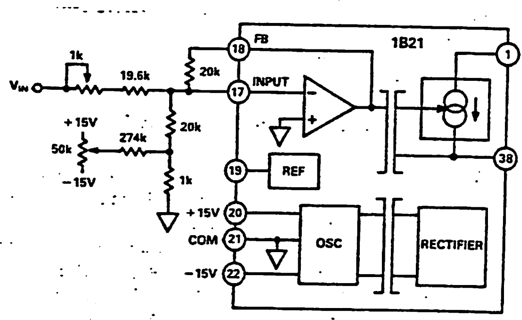


Figure 2. Offset and Span Adjustment

TC Considerations of External Resistors: The specifications for gain and offset temperature coefficient (TC) for the 1B21 exclude the effects of external components. The total gain TC for the circuit in Figure 1 is:

Gain TC = 1B21 Gain TC + (Tracking TC of R_F and R_I)

The offset TC is also affected by the thermal stability of the internal voltage reference and its contribution is:

$$\text{Ref TC} = (V_{REF})(R_F/R_O)(4mA/V)(\text{TC of } V_{REF} + \text{Tracking TC of } R_F \text{ and } R_O)/1 \times 10^6$$

Total Offset TC = 1B21 Offset TC + Ref TC

Specifically using R_F, R_I and R_O from Case 3 in Table I, with absolute TCs of ±25ppm/°C we get:

$$\text{Gain TC} = 50 + (25+25) = 100\text{ppm/°C}$$
$$\text{Offset TC} = 300 + (6.4V)(20k/128k)(4mA/V)(20+25+25)/1 \times 10^6 = \pm 580\text{nA/°C}$$

Similarly, when using a resistor network with a tracking spec of ±5ppm/°C, the total gain TC would be ±55ppm/°C and the total offset TC would be ±400nA/°C.

APPLICATIONS

Output Protection: In many industrial applications it may be necessary to protect the current output from accidental shorts to ac line voltages in addition to high common-mode voltages and short circuits to ground. The circuit shown in Figure 3 can be used for this purpose. The maximum permissible load resistance will be lowered by the fuse resistance (typically 8Ω) when protection circuitry is utilized.

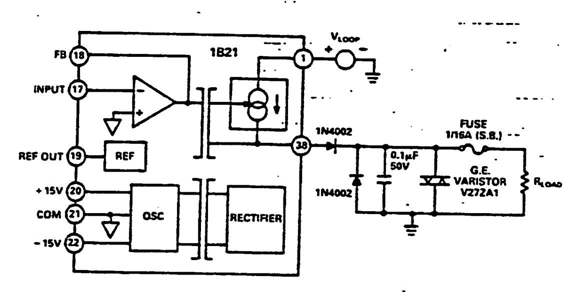


Figure 3. Output Protection Circuitry

Low Drift Input Network: Figure 4 shows a configuration suitable for applications where errors have to be minimized over a wide temperature range. A temperature tracking network such as a 50k Beckman (PN 698-3R50KD) can be used to implement both offset and gain for either 0 to 20mA or 4mA to 20mA current loops. For 0-10V signals either IN1 or IN2 can be used for input. For 0-5V signals, jumper IN1 to IN2. Similarly, for 4mA to 20mA operation the 4mA node should be jumpered to OFFSET, while for 0 to 20mA it should be tied to COM.

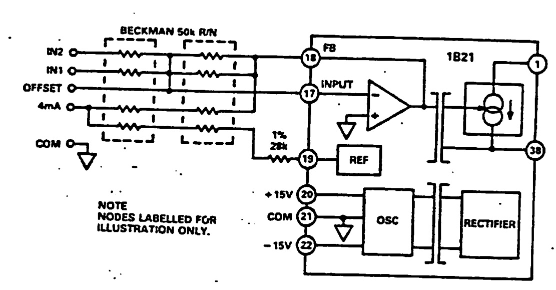


Figure 4. Low Tempco Resistor Network Configuration

Multiloop Isolation: Multiple 1B21s can be connected to a single loop supply in parallel as shown in Figure 5. The amperage of the loop supply should be sufficient to drive all the loops at full-scale output.

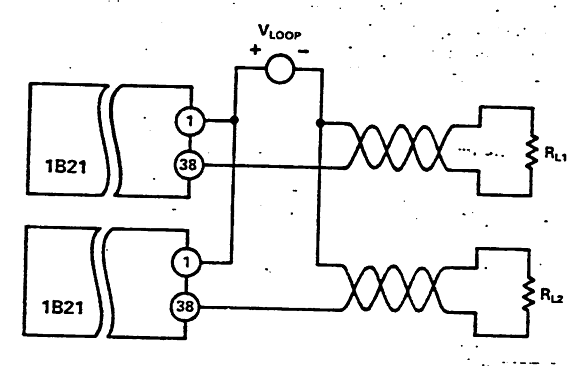


Figure 5. Multiple IB21s with Single Loop Supply

ERKLÄRUNG

Ich versichere, daß ich die Arbeit ohne fremde Hilfe angefertigt und mich anderer als der von mir angegebenen Hilfsmittel nicht bedient habe

Dieburg, den 27.12.1984

Wolfgang Schmal
(Unterschrift)

